Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Programa de Pós-Graduação em Informática Aplicada

Uma Estratégia baseada em Modelos para a Quantificação do Impacto da Disponibilidade no Fluxo Energético de *Data Centers*

Thiago Valentim Bezerra

Recife

Setembro de 2019

Thiago Valentim

Uma Estratégia baseada em Modelos para a Quantificação do Impacto da Disponibilidade no Fluxo Energético de *Data Centers*

Orientador: Prof. Dr. Gustavo Rau de Almeida Callou

Dissertação de mestrado apresentada ao Curso de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do grau de Mestre em Informática Aplicada.

Recife

Setembro de 2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

B574e Bezerra, Thiago Valentim
Uma estratégia baseada em modelos para a quantificação do impacto da disponibilidade no fluxo energético de data centers / Thiago Valentim Bezerra. - 2019.
92 f. : il.

Orientador: Gustavo Rau de Almeida Callou.
Inclui referências.

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2019.

1. Disponibilidade. 2. Modelo de fluxo de energia. 3. Centro de dados. 4. Petri, Redes de. 5. Petri coloridas, Redes de. I. Callou, Gustavo Rau de Almeida, orient. II. Título

CDD 004

Thiago Valentim

Uma Estratégia baseada em Modelos para a Quantificação do Impacto da Disponibilidade no Fluxo Energético de *Data Centers*

Orientador: Prof. Dr. Gustavo Rau de Almeida Callou

Aprovada em: 30 de Setembro de 2019.

BANCA EXAMINADORA

Prof. Dr. Gustavo Rau de Almeida Callou
Universidade Federal Rural de Pernambuco

Prof. Dr. Gabriel Alves de Albuquerque Junior
Universidade Federal Rural de Pernambuco

Prof. Dr. Bruno Costa e Silva Nogueira
Universidade Federal de Alagoas

À,
Meus filhos Rafael, Emanuel e Pedro (In Memoriam), Minha esposa, Meus pais, Meu irmão, meus amigos de toda a vida

Agradecimentos

Agradeço primeiramente a Deus, pois sem Ele eu nada seria e por me dar a persistência necessária para não desistir diante das dificuldades encontradas no caminho. A minha amada esposa, Milena Rodrigues, por ter a compreensão e a paciência do mundo durante esse período. Ela me motivou a permanecer firme quando as coisas ficaram difíceis. Aos meus filhos: Pedro (In Memorian), Rafael e Emanuel, pois, eles são os motivos de seguir em frente, mesmo com as dificuldades da vida. Todos os dias quando acordo, são meus primeiros pensamentos. Aos meus pais Francisco e Damares, pois sem eles não chegaria aqui. Ao meu irmão Filipe, aos meus amigos de mestrado que fiz durante essa jornada. Agradeço ao meu orientador, o Prof. Dr. Gustavo Callou, por compartilhar todo o conhecimento necessário para que este trabalho pudesse chegar ao fim, por sempre estar presente e, principalmente, pela paciência e confiança em mim. Agradeço ao Prof. Dr. Gabriel alves, pelos direcionamentos iniciais sobre CPN, fundamental para o andamento deste trabalho. Agradeço a todos os professores do BSI que durante a graduação passaram todo o conhecimento. Agradeço a todos os meus amigos, que conheci na ETE Professor José Luiz de Medonça, por estarem sempre ao meu lado me dando forças pra seguir adiante. Agradeço a todos que amo, e que me ajudaram nessa tão sonhada conquista.

Resumo

Com o aumento da demanda por maior poder computacional, a procura por serviços hospedados em ambientes de nuvens computacionais vem aumentando. Sabe-se, por exemplo, que em 2018 mais de 4 bilhões de pessoas fizeram acessos diários a estes serviços por meio da Internet, valor correspondente a mais da metade da população mundial. Para prover suporte a tais serviços, estas nuvens são disponibilizadas por grandes *data centers*. Esses sistemas acabam por serem responsáveis pelo consumo cada vez maior de energia elétrica, tendo em vista a crescente quantidade de acessos, aumentando a demanda por uma maior capacidade de comunicação, processamento e por alta disponibilidade. Uma vez que esta energia elétrica nem sempre é obtida a partir de recursos renováveis, a incessante busca por serviços de nuvens pode ter como efeito colateral um relevante impacto ambiental. Neste contexto, este trabalho propõe uma estratégia integrada e dinâmica que demostra o impacto da disponibilidade dos equipamentos que compõem a arquitetura de *data centers* no consumo energético. Para isso, foi utilizado a técnica de modelagem redes de Petri coloridas, responsável pela quantificação do custo, impacto ambiental e disponibilidade da infraestrutura de energia elétrica dos *data centers* em análise. Tais modelos propostos são suportados pelo ferramental desenvolvido, onde projetistas de *data centers* não precisam conhecer o formalismo de redes de Petri para computar as métricas de interesse. Dois estudos de caso foram propostos para mostrar a aplicabilidade dos modelos e da estratégia proposta. Resultados significativos foram obtidos, mostrando um aumento na disponibilidade do sistema de 100%, com praticamente o mesmo custo operacional e o mesmo impacto ambiental.

Abstract

As demand for higher computing power increases, demand for services hosted in cloud computing environments is increasing. For example, it is known that in 2018 over 4 billion people made daily access to these services through the Internet, which is more than half of the world's population. To support such services, these clouds are made available by large data centers. These systems are responsible for the increasing consumption of electricity, given the increasing number of accesses, increasing the demand for greater communication capacity, processing and high availability. Since this electricity is not always obtained from renewable resources, the relentless pursuit of cloud services can have a significant environmental impact. In this context, this paper proposes an integrated and dynamic strategy that demonstrates the impact of the availability of data center architecture equipment on energy consumption. For this, we used the technique of modeling colored Petri nets, responsible for quantifying the cost, environmental impact and availability of the electric power infrastructure of the data centers under analysis. Such proposed models are supported by the developed tooling, where data center designers do not need to know the Petri net formalism to compute the metrics of interest. Two case studies were proposed to show the applicability of the models and the proposed strategy. Significant results were obtained, showing a 100% increase in system availability, with virtually the same operating cost and environmental impact.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de acrônimos

CGSPN	Colored generalized stochastic Petri net
CPN	Colored Petri Net
CRM	Customer Relationship Management
CTPNs	Colored timed petri nets
EFM	Modelo de Fluxo de Energia
FMECA	Modelos de falha, efeitos e análise de criticidade
HCPN	Hierarchical Colored Petri Net
IAAS	Infrastructure as a Service
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
PAAS	Platform as a Service
PDU	Power Distribution Units
RBD	Reliability Block Diagram
RP	Redes de Petri
SAAS	Software as a Service
SAN	Storage Area Network
SDT	Step Down Transformers
SPN	Stochastic Petri Net

STS	Static Transfer Switch
TI	Tecnologia da Informação
UPS	Uninterruptible Power Supplies

Capítulo 1

Introdução

Os serviços em nuvem cada vez mais difundidos e utilizados pelos usuários da grande rede demandaram um aumento na capacidade de processamento e armazenamento de dados dos *data centers*. Por exemplo, mais de dois bilhões de pessoas acessaram a Internet em 2011, à uma década esse volume era de 250 milhões [1]. Em 2018, mais de 4 bilhões de pessoas acessaram a Internet, ou seja, mais da metade da população mundial [2].

A computação em nuvem, através do uso da tecnologia baseada na Internet para as transações comerciais, tem sido usada de forma crescente pelas maiores empresas de comércio eletrônico, como também pela população mundial [3]. Principalmente com o avanço da tecnologia com novos dispositivos com acesso à Internet e aplicações baseadas nesses serviços [4].

Os tipos de serviços em nuvem fornecidos pelas empresas que trabalham com esse paradigma, são [5]:

- IaaS: Infraestrutura como serviço, ou seja, uma infraestrutura completa de equipamentos;
- Paas: Plataforma como serviço, ou seja, fornecem um ambiente de desenvolvimento de aplicações;
- Saas: Software como serviço, ou seja, fornecem aplicações completa para uso, como

exemplo: aplicações de CRM (*Customer Relationship Management*).

Para que os serviços, como: armazenamento, *middleware*, softwares de colaboração e recursos de banco de dados atingissem altos níveis de disponibilidade exigidos pela computação em nuvem, foi necessário que os *data centers* evoluíssem. Para tal, foram implementados estratégias redundantes de componentes [6].

Desta forma, o aumento da disponibilidade melhora os serviços oferecidos, porém, esse aumento impacta diretamente na sustentabilidade. Sobretudo porque um *data center* não é construído apenas por componentes de TI, mas também por toda a infraestrutura de energia e refrigeração, que consomem uma parte significativa de energia.

No entanto, a redundância leva a equipamentos extras, sendo necessário mais energia, o que pode resultar em um impacto negativo na sustentabilidade e no custo do sistema. Portanto, as preocupações sobre confiabilidade, sustentabilidade e custo dos sistemas de *data center* estão em foco tanto pela comunidade acadêmica quanto pela sociedade.

Esse crescimento considerável traz um consumo de energia que corresponde por cerca de 2% da geração atual de energia dos EUA [7]. Esse crescente aumento no consumo da energia, pode trazer um grande efeito colateral, sobretudo, no meio ambiente. Os *data centers* de grande porte têm uma importância neste contexto, visto que para os *data centers* funcionarem são necessários diversos equipamentos que consomem energia. A estimativa é que os *data centers* usarão entre 3% e 13% da eletricidade global em 2030 [8].

1.1 Motivação e Justificativa

É interessante que os projetistas dos *data centers* consigam estimar os custos de operação antes mesmo da implantação do sistema real. Os *data centers* chegam a consumir de 100 a 200 vezes mais eletricidade do que os escritórios de empresas convencionais [9]. Além de tudo, os valores do consumo de energia dos *data centers* dobram a cada cinco anos [10]. Em certas ocasiões os custos de energia podem passar dos custos de aquisição dos componentes [11].

O modelo proposto nesse trabalho visa auxiliar os projetistas, ao propor uma estratégia integrada e dinâmica utilizando a técnica de modelagem das redes de Petri coloridas (CPN) para computar o impacto da disponibilidade na infraestrutura de TI do *data center*, ou seja, estimar com antecedência o consumo energético dessas arquiteturas. Além do mais, o presente trabalho também quantifica custo e disponibilidade, por exemplo, para dar suporte aos projetistas dos *data centers*.

Desta forma, os projetistas serão capazes de estimar o impacto da disponibilidade no fluxo energético, isto é, quando um equipamento da arquitetura apresentar uma inoperância o modelo computa o seu consumo energético. Além disso, se o modelo apresentar caminhos redundantes para transmissão da energia elétrica e um dos caminhos apresentar inoperância, o modelo irá transferir a energia pelo caminho onde os equipamentos estejam ativos. Ademais, o modelo proposto também realiza a verificação da potência máxima que cada componente pode fornecer.

Trabalhos como Callou et al. [12] tiveram como objetivo a proposição de um modelo, denominado EFM (*energy flow model*), que verifica e garante as restrições de capacidade de potência máxima que cada equipamento pode fornecer. Porém, o EFM não foi proposto pensando em calcular o impacto, de forma dinâmica, da disponibilidade de um dispositivo no fluxo de energia do *data center*. A partir dos resultados de disponibilidade calculados fora do EFM, o modelo é capaz de estimar a eficiência da arquitetura proposta e o impacto ambiental decorrente.

1.2 Objetivos

Esta dissertação propõe uma estratégia integrada e dinâmica que demostra o impacto da disponibilidade dos equipamentos que compõem a arquitetura de *data centers* no consumo energético. Para isso, modelos CPN são propostos para quantificar custo, impacto ambiental e disponibilidade da infraestrutura da energia elétrica de *data center*. A estratégia adotada inicia com o entendimento da arquitetura elétrica de um *data center*, seguida pelo entendimento do funcionamento da disponibilidade. Logo após, pela criação de modelos em CPN.

Será utilizado o ambiente Mercury [13], ferramenta que fornece suporte ao EFM, SPN, RBD, para a validação dos modelos. Em seguida, o modelo em CPN irá simular as arquiteturas levando em consideração sua disponibilidade. O modelo terá como entrada alguns equipamentos (com diferentes parâmetros, por exemplo, MTTF, MTTR, custo e eficiência energética) pertencentes a infraestrutura elétrica de *data center*.

De forma mais específica os objetivos deste trabalho são:

- Propor um modelo (ex., em CPN) que mostre o impacto da indisponibilidade dos equipamentos que compõem a arquitetura de *data centers* no consumo energético destas arquiteturas;
- Propor uma estratégia para relacionar o modelo de fluxo de energia com o modelo de disponibilidade. Essa estratégia fará com que a falha no modelo de disponibilidade em tempo de execução impacte o modelo de fluxo de energia;
- Avaliar cenários através de estudos de casos para demostrar a aplicabilidade dos modelos propostos.

Dois estudos de caso serão apresentados para validar os modelos propostos neste trabalho. A avaliação é com base em cinco arquiteturas básicas de *data center*. O primeiro estudo de caso valida os resultados obtidos em CPN com outras técnicas de modelagem. O segundo estudo de caso irá avaliar o consumo energético das arquiteturas em tempo de execução do sistema. A comparação com essas técnicas de modelagem comprovará a eficiência e eficácia do modelo proposto neste trabalho.

1.3 Estrutura da dissertação

O restante desta dissertação está organizado como segue. O Capítulo 2 apresenta os conceitos básicos para um melhor entendimento deste trabalho, iniciando com conceitos de redes de Petri coloridas, seguido pela apresentação do modelo de fluxo de energia (EFM) e, finalmente, termina com uma breve explicação sobre disponibilidade. O Capítulo 3 mostra trabalhos relacionados a essa pesquisa. O Capítulo 4 apresenta a metodologia adotada

para a estimativa das métricas de custo, consumo de energia elétrica, disponibilidade e o critério de parada da simulação adotada na avaliação do modelo proposto. O Capítulo 5 apresenta os modelos propostos em redes de Petri coloridas. O Capítulo 6 apresenta um estudo experimental para mostrar a aplicabilidade dos modelos e, também, fazer a validação desses modelos. Por último, o Capítulo 7 conclui e faz sugestões sobre os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta, brevemente, os conceitos e conhecimentos necessários para um melhor entendimento sobre este trabalho. O capitulo inicia com uma visão geral do processo de simulação e, em seguida, continua com a apresentação das infra-estruturas típicas de *data centers*. Posteriormente, são apresentados os conceitos de disponibilidade e exergia que são amplamente utilizadas nas métricas quantificadas neste trabalho. O capítulo segue com a apresentação das redes de Petri elementares e, em seguida, é apresentada a rede de Petri colorida. Por fim, um explicação sobre o modelo EFM é conduzida.

2.1 Processo de simulação

A execução de uma simulação reproduz um modelo representado por um sistema. Nesse contexto, existem dois tipos de sistemas: terminal e não terminal. O sistema terminal, também chamado de sistema transitório, é aquele em que há estado inicial e final bem determinado. O não terminal, também conhecido de sistema estacionário, consiste em um sistema que a simulação é finalizada através de uma avaliação estatística definido por um critério de parada ao invés de um evento que poderia acontecer. Foi adotada uma abordagem de simulação estacionária neste trabalho.

A Figura 2.1 descreve um processo de simulação comum [14]. A simulação começa no

programa principal, que chama a rotina de inicialização. A rotina de inicialização define o *clock* da simulação para "0" (variável que indica o valor atual do tempo simulado), inicializando os contadores (variáveis usadas para armazenar informações estatísticas sobre o desempenho do sistema e para cada transição que é capaz de disparar).

Depois, o programa principal chama a rotina de temporização que determina qual será o próximo tipo de evento (a transição que é disparada) e avança o *clock* de simulação. Em seguida, o programa principal chama a rotina de eventos, na qual o estado do sistema e os contadores estatísticos são atualizados, próximos eventos são gerados e adicionados à lista de eventos. Então, é determinado se a simulação deve ser finalizada ou não, de acordo com a avaliação do critério de parada. Depois de terminar a simulação, os resultados das estimativas são mostrados.

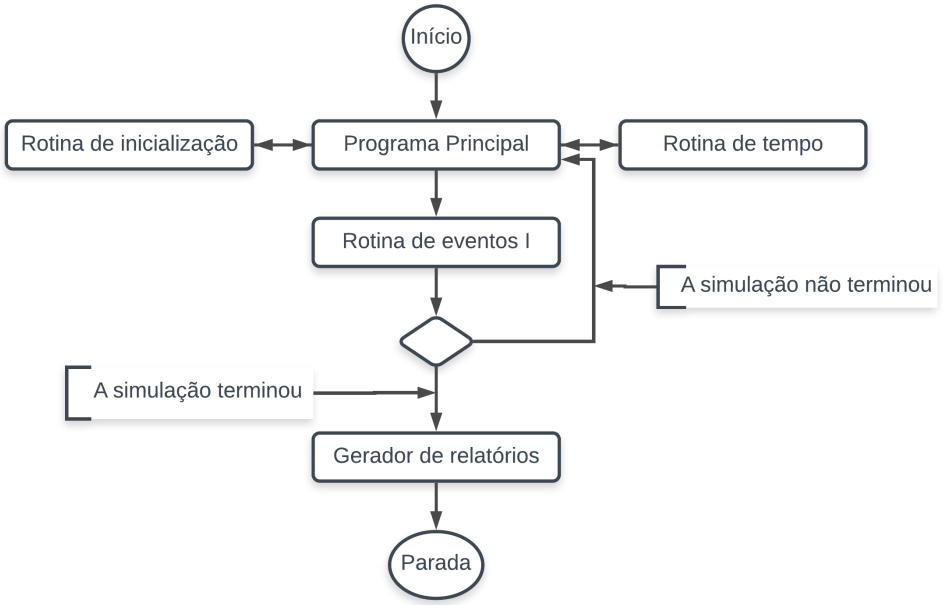


Figura 2.1: Diagrama do processo de simulação

2.2 Data Centers

O *data center* comum é definido como sendo o ambiente de alto índice de processamento, armazenamento e onde estão instalados os aplicativos necessários para suportar os negócios

empresariais. O planejamento da infraestrutura do *data center* é essencial para o bom funcionamento dos serviços de TI. A modelagem adequada do projeto de infraestrutura do *data center* é importante para o desempenho, resiliência, disponibilidade e escalabilidade [15].

É necessário que o *data center* tenha altos níveis de confiabilidade para melhor entregar o seu serviço. Para que isso aconteça, é interessante que a sua infraestrutura seja composta por: (i) fontes alternativas de energia, quando ocorre a falha de uma concessionária de energia elétrica o serviço não pare; (ii) equipamentos de redes com alta disponibilidade e redundantes, onde se um desses equipamentos parar de funcionar o outro assume o serviço; (iii) caminhos de dados redundantes para ampliar os trajetos alternativos.

A Figura 2.2 descreve as diferentes camadas necessárias para a implementação de um *data center*. Vai da estrutura física até o último serviço de TI fornecido pela arquitetura. As camadas tem domínios diferentes, isso mostra uma visão geral e um entendimento dos componentes que compõem um *data center*.

Figura 2.2: Infraestrutura típica de um *Data Center*

Um *data center* tem a finalidade de agrupar equipamentos de TI que disponibiliza determinado serviço de TI por meio da rede. Uma descrição básica de um *data center* começa com o seu espaço físico, usado para abrigar os componentes de TI, por exemplo. Podendo ser uma sala dentro de um edifício ou mesmo em edifício de uso exclusivo para este serviço [16].

A infraestrutura de um *data center* geralmente, garante que os componentes de TI funcionem de forma confiável e inclui: (i) infraestrutura de TI; (ii) infraestrutura de resfriamento; e (iii) infraestrutura de energia [16].

A infraestrutura de resfriamento é composta por uma variedade de equipamentos, como: compressores mecânicos (condicionadores de ar), resfriadores e torres de resfriamento [17].

A infraestrutura elétrica tem a função de distribuir a eletricidade aos componentes de TI, e garante que a eletricidade chegue sem instabilidade aos equipamentos, hospedados em *racks*. Essa infraestrutura inclui os fornecimentos de fontes de alimentação alternativas, fornecidas por fontes de alimentação ininterruptas (UPS) que contém baterias para suprir as faltas de energia da rede elétrica [18].

A partir da concessionária de energia elétrica, a energia normalmente passa por transformadores (SDT), chaves de transferência, fontes de alimentação ininterruptas (UPS), unidades de distribuição de energia (PDU) e, por fim, pelas réguas de energia.

A infraestrutura de TI tem a finalidade de fornecer os serviços de TI que podem ser separados em funções de rede que incluem o hardware físico adotado para transmitir dados eletronicamente, como *switches*, *hubs*, roteadores, *bridge* e *gateways*; armazenamento que são conectados por meio de uma rede de área de armazenamento (SAN); e processamento de dados [19]. Embora os serviços de armazenamento, rede e processamento de dados sejam usados de forma dedicadas, a peça principal continua sendo os servidores de TI. Eles podem fornecer e controlar o acesso a esses tipos de recursos.

Os servidores de TI executam os sistemas operacionais que através deles, irão executar softwares como: serviços de gerenciamento e virtualização. Essas aplicações irão fornecer serviços aos usuários.

2.3 Disponibilidade

A disponibilidade é uma métrica de desempenho que utiliza propriedades de confiabilidade e capacidade de manutenção de um componente ou sistema [20]. A disponibilidade é definida como a probabilidade de um conjunto de serviços estarem acessíveis por um deter-

minado período de tempo [21]. Ou seja, é expressa pela relação entre o tempo ativo *(uptime)* do sistema e o tempo inativo *(downtime)* do sistema, conforme mostrado na Equação 2.1.

$$A = \frac{E[Uptime]}{E[Uptime] + E[Donwtime]} \tag{2.1}$$

Disponibilidade Inerente

É a disponibilidade quando apenas é considerado o tempo de atividade/inatividade do sistema. Dessa forma, retira-se o tempo de logística das peças (quando é necessário) e a inatividade da manutenção preventiva. Também é excluído o tempo de inatividade quando o sistema está passando pela manutenção corretiva [20].

Para calcular a disponibilidade inerente de um serviço, é necessário conhecer o tempo em que o serviço está disponível e o tempo do seu reparo após a falha. A disponibilidade (A) pode ser calculada a partir do tempo médio de falha (MTTF) e o tempo médio de reparo (MTTR) [22]. A Equação 2.2 mostra a forma de se calcular a disponibilidade (A), é pela razão entre o *uptime* (serviço disponível) do sistema e a soma do *uptime* com o *downtime* (tempo de indisponibilidade do serviço) do sistema.

$$A = \frac{MTTF}{MTTF + MTTR} \tag{2.2}$$

2.3.1 Cálculo da disponibilidade em relação a sistemas série/paralelo

Quando se pretende calcular a disponibilidade de um sistema é necessário conhecer a organização entre os componentes que compõem o sistema em análise. Em um sistema em série, quando um único componente falhar, todo o sistema não estará mais operacional. Assumindo um sistema com n componentes independentes, a confiabilidade (disponibilidade

instantânea) é obtida conforme a Equação 2.3.

$$P_s = \prod_{i=1}^{n} P_i \tag{2.3}$$

onde P_s é a confiabilidade do sistema e P_i é a confiabilidade do componente.

Em um sistema em paralelo, se um único componente estiver operacional, todo o sistema também estará operacional. Assumindo um sistema com n componentes independentes, a confiabilidade (disponibilidade instantânea) é obtida conforme a Equação 2.4.

$$P_s = 1 - \prod_{i=1}^{n} (1 - P_i) \tag{2.4}$$

onde P_s é a confiabilidade do sistema e P_i é a confiabilidade do componente.

2.4 Exergia

Exergia é definida como a parte da energia que será convertida em trabalho útil. O conceito de exergia é um resultado direto da segunda lei da termodinâmica que trata da transferência de energia térmica, onde nesse processo sempre ocorre uma perda [23, 24]. Como resultado, a exergia consumida ou destruída (ou simplesmente, exergia, para esse trabalho) mostra o quanto eficiente o sistema é, com relação a perda de energia. A exergia é calculada como o produto de energia por um fator de qualidade de acordo com a Equação 2.5.

$$Exergia = Energia \times F \tag{2.5}$$

2.5 Redes de Petri

O conceito de redes de Petri foi primeiramente apresentado por Carl Adams Petri [25] na sua dissertação de doutorado na Universidade de Darmstadt, na Alemanha. É uma forma de representar graficamente e matematicamente diversos tipos de processos com características distintas como: sistemas com recursos paralelos, concorrentes, assíncronos e sistemas não determinísticos [26]. Dessa forma é possível visualizar o estado do sistema em determinado momento como também suas transições.

A partir do primeiro formalismo, outras representações foram propostas permitindo extensões que representassem características que não foram abordadas nos primeiros modelos. Dessa forma, a rede de Petri foi adaptada e estendida em diversos seguimentos, nas quais redes temporizadas [27], estocásticas [28], de alto nível (CPN) [29] e orientadas a objetos [30].

As redes de Petri são compostas pelos elementos a seguir:

- Lugares: tem como função o armazenamento de *tokens* os quais são removidos e adicionados à medida que as transições são disparadas. Representam os elementos passivos da rede, pode acontecer de conter mais de um (ou nenhum) *token* dentro do lugar.
- Token: o *token* é representado para marcar a atual condição que poderá ser mantida ou pode ser movida pela ocorrência do disparo das transições;
- Transição: são os elementos ativos da rede, em outras palavras, as ações realizadas pelo sistema. O disparo de uma transição faz com que *tokens* sejam consumidos dos lugares de entrada da transição e gerados nos lugares de saídas da transição. Uma transição se encontra habilitada para disparar se o número de *tokens* presentes nos lugares de entrada da transição são iguais ou superiores ao peso dos arcos de entrada da transição.
- Arco: o arco faz a conexão dos elementos principais (lugares e transições) que mostram a direção do fluxo. Os arcos conectam apenas os componentes principais intercalando lugar e transição, ou seja, um arco nunca se une a dois lugares ou duas transições.

Os elementos básicos de uma rede de Petri estão representados na Figura 2.3.

Figura 2.3: Elementos da Rede de Petri

A representação formal de um modelo em rede de Petri (PN) é a 5-tupla $PN = \{P, T, F, W, M_0\}$ [26], onde:

(i) $P = \{p_1, p_2, ..., p_m\}$ é o conjunto finito de lugares;

(ii) $T = \{t_1, t_2, ..., t_n\}$ é o conjunto finito de transições;

(iii) $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos;

(iv) $W : F \rightarrow \{1, 2, 3, ...\}$ é a função de atribuição de peso aos arcos;

(v) $M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ é a marcação inicial.

O exemplo da Figura 2.4 representa a modelagem do funcionamento de um interruptor de luz em rede de Petri. Nesse modelo, os lugares representam o estado em que pode se encontrar o funcionamento do interruptor (*Ligado* ou *Desligado*) e as transições representam as ações que alteram o estado do interruptor (*Ligar* ou *Desligar*). Um *token* marca o estado do modelo no lugar correspondente à situação atual do interruptor como descrito na Figura 2.4 (a). Considerando que *Ligado* é o estado atual do modelo, transição *Desligar* fica habilitada para disparar. Uma vez disparada esta transição, o modelo passa do estado *Ligado* (ver Figura 2.4 (a) para o estado *Desligado* (ver Figura 2.4 (b)).

2.6 Redes de Petri Coloridas

Entre as diversas extensões de redes de Petri que foram propostas é interessante ressaltar o modelo de alto nível de Jensen, a rede de Petri colorida (CPN) [31, 32]. As redes de

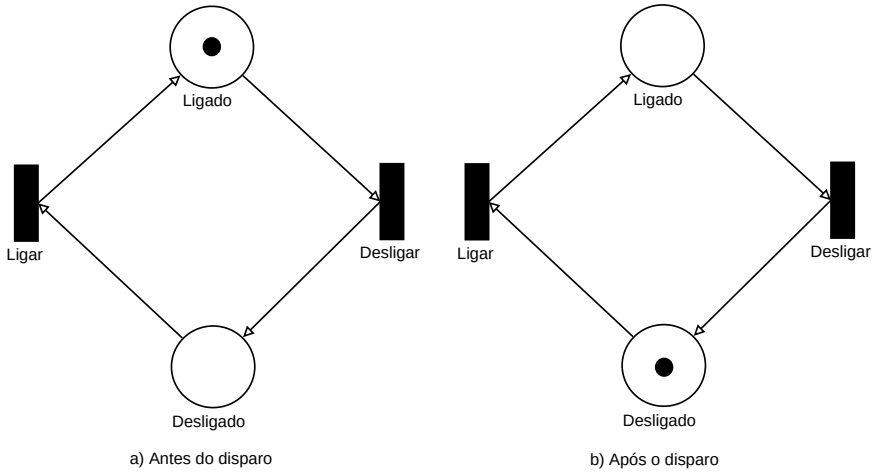


Figura 2.4: Exemplo do funcionamento de um interruptor modelado em rede de Petri .

Petri coloridas (CPN) são uma extensão das redes de Petri [26] por poderem definir tipos complexos para os *tokens* e por terem uma linguagem própria de programação (CPN ML) associada [33]. Sendo assim, as CPNs são consideradas como redes de alto nível e permitem a modelagem de um problema com diversos níveis de hierarquia. Todos esses fatores permitem a simplificação dos modelos que ficam menores e a versão de mais alto nível da hierarquia pode facilitar o entendimento do sistema modelado.

Da mesma forma em RP, os lugares são representados graficamente por elipses, a transição por retângulos e o arco por setas. Além disso, uma CPN temporizada foi adotada nesta pesquisa. A principal diferença entre os modelos CPN com tempo e os modelos CPN sem tempo é que a transição do modelo somente será habilitada quando o tempo da simulação for igual ao tempo estabelecido no *token* [32].

Nas CPNs, modelos com uma estrutura de hierarquia podem ser criados, permitindo a modelagem a partir de diferentes níveis de abstrações [34]. O processo de entrada das transições podem demandar valores de acordo com a função adotada e gerar um *token* com valor específico [35].

CPN introduziu diversas propriedades que transformam esse formalismo em uma valiosa linguagem para a especificação e análise de diversos sistemas. Entre essas propriedades, é possível destacar:

(i) Semântica bem definida: CPNs têm uma semântica bem definida que as tornam capa-

zes de representar sistemas complexos;

(ii) Modelo em hierarquia: é possível construir uma CPN grande e complexa, relacionando CPNs menores a cada um, de maneira bem estruturada. Esta propriedade hierárquica é semelhante a sub-rotinas, procedimentos e módulos de linguagens de programação;

(iii) Conceito de tempo: as CPNs podem ser construídas de forma a aceitar o conceito de tempo;

(iv) Simulações interativas: nas simulações de CPN, é possível disparar um determinado número de transições e verificar seus resultados instantaneamente;

(v) Inscrições: as CPNs permitem associar inscrições a componentes CPN, como lugares, arcos e transições, a fim de melhorar suas funcionalidades;

(vi) Processo de redução: é possível aplicar um processo de redução onde um novo modelo simplificado de CPN pode ser obtido preservando propriedades determinadas.

A rede de Petri colorida baseia-se nas seguintes propriedades:

- Conjunto de cores: uma função que descreve o conjunto de coleções de elementos do mesmo dado.
- Conjuntos de operações: operações básicas estão definidas nos conjuntos de cores, são elas: adição, multiplicação escalar, comparação modular e subtração.
- Tipo de operador: o tipo de operador mapeia a variável ou expressão *exp* em tipos válidos.
- Operador de um conjunto de variáveis: o operador de um conjunto de variáveis, representa o conjunto de variáveis em uma expressão *exp*.
- Token: o *token* representa um valor (literal) que pode ser um tipo de dado simples ou composto.
- Elemento de ligação: Um elemento de ligação é um par (t, b) onde t $\in$ T e b $\in B_{(t)}$, onde $B_{(t)}$ denota o conjunto de todas as ligações para t.

- Marcação: consiste em um número de *tokens* posicionados (distribuídos) nos lugares. Cada *token* carrega um valor(*color set*), que pertence ao tipo do lugar em que o *token* se encontra. Os *tokens* presentes em um determinado lugar são chamados de marcação do lugar [36].

- Transição Habilitada: Quando uma transição é habilitada, pode ocorrer o disparo. Quando ocorre o disparo o *token* é removido do lugar conectado a um arco de entrada (lugar de entrada) e adiciona o *token* no lugar conectado ao arco de saída (lugares de saída), alterando assim a marcação (estado) da CPN [36].

A Representação formal de um modelo em rede de Petri (RP) é a nona-tupla $CPN = (\sum, P, T, A, N, C, G, E, I)$ [26], onde:

(i) $\sum$ é um conjunto não vazio, chamado de conjuntos de cores;

(ii) P é um conjunto finito de lugares;

(iii) T é o conjunto finito de transição;

(iv) A é um conjunto finito de arcos de tal forma que $\mathrm{P} \cap \mathrm{T} = \mathrm{P} \cap \mathrm{A} = \mathrm{T} \cap \mathrm{A} = \varnothing$;

(v) N é uma função nó definido de A para $\mathrm{P} \times \mathrm{T} \cup \mathrm{T} \times \mathrm{P}$;

(vi) C é um conjunto de cores definido a partir de P em $\sum$;

(vii) G é uma função de guarda definida de T em expressões tais que: $\forall t \in T : [Type\,(G\,(t)) = Bool \wedge Type\,(Var\,(G\,(t))) \subseteq \sum]$, onde $Bool \in \{verdadeiro, falso\}$;

(viii) E é uma função de arco definida de A em expressões tais que: $\forall a \in A : [Type\,(E\,(a)) = C\,(p\,(s))_{MS} \wedge Type\,(Var\,(E\,(a))) \subseteq \sum]$, onde $p\,(a)$ é o espaço de $N\,(a)$ e C_{MS} observa o conjunto de todos os multi-conjuntos sobre C;

(ix) I é uma função de inicialização definido de P em expressões fechadas de tal forma que $\forall_a \in P : [Type\,(I\,(p)) = C\,(p\,(s))_{MS}]$.

onde:

$Type\,(expr)$ indica os tipos de expressões;

$Var\,(expr)$ indica o conjunto de variáveis da expressão;

$C\,(p)_{MS}$ indica um conjunto múltiplo sobre $C\,(p)$

indica o conjunto de variáveis da expressão;

Depois de analisar a definição formal da CPN, o leitor deve concluir que:

O conjunto de cores - item (i) determina os valores de dados, operações e funções que podem ser adotadas nas expressões (i, e: expressões de arco, guarda e expressões de inicialização).

Lugares, transições e arcos - item (ii), (iii), (iv) - são descritos por três conjuntos P, T e A que são finitos e separados por pares $P \cap T = P \cap A = T \cap A = \varnothing$.

A função do nó - item (v) - mapeia cada arco em um par onde o primeiro elemento é o nó de origem e o segundo o nó de destino. É importante ressaltar que ambos os elementos devem ser do tipo diferente (i, e: lugar e transição).

A função de *colour* C - item (vi) - mapeia cada lugar, p, para um tipo $C(p) \in \sum$. Isto significa que cada *token* em p deve ter um valor de dados que pertence a $C(p)$.

A função guarda G - item (vii) - mapeia cada transição, t, para uma expressão booleana, onde todas as variáveis possuem tipos que pertencem a S.

A função de expressão de arco E - item (viii) - mapeia cada arco, a, em uma expressão do tipo $C(p)_{MS}$. Isso significa que cada expressão do arco deve ser avaliada em vários conjuntos sobre o tipo de lugar adjacente, p.

A função de inicialização I - item (ix) - mapeia cada lugar, p, em uma expressão fechada que seja do tipo $C(p)_{MS}$. Uma expressão fechada é uma expressão sem variáveis que podem ser avaliadas em todas as ligações e todas as avaliações fornecem o mesmo valor.

2.6.1 Exemplo de CPN

Para melhor entendimento do funcionamento das CPNs, a Figura 2.5 apresenta um modelo de um simples interruptor de luz, composto por duas transições e dois lugares. Quando o *token* está no lugar *ON* o seu valor é *light=1*, representando que o interruptor está com a luz ligada. Com o disparo da *transition_1* o *token* vai para o lugar *OFF*, modificando o seu valor para *light*=0, indicando que a luz está desligada. Já com o disparo da *transition_2*, o *token* volta para o lugar *ON*, representando que a luz está ligada (valor do token light=1).

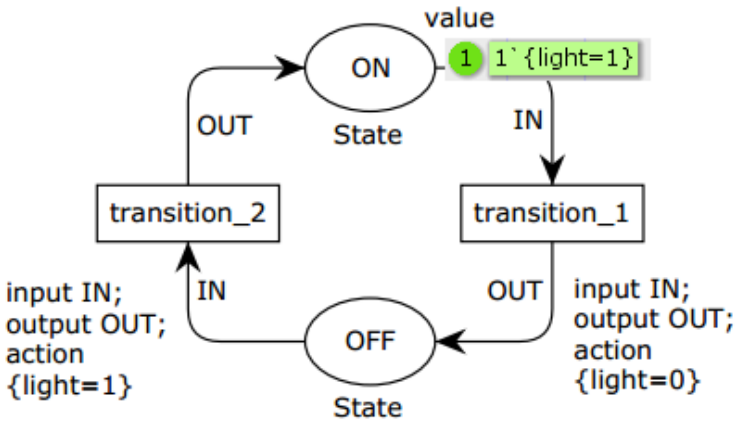


Figura 2.5: Representação de um interruptor de luz

A Figura 2.6 mostra as declarações do modelo da Figura 2 que representa um interruptor de luz. Neste modelo, existe uma declaração do conjunto de cor *State* que é um registro (*record*) denominado por *light*, do tipo inteiro. Nesse exemplo da Figura 2.5, o *token* inicia com o valor de *light=1*. Nas CPNs, todos os arcos devem ter o seu conjunto de cores associado para indicar qual tipo de *token* que pode passar pelo arco. Nesse exemplo, os tipos dos arcos são representados pelas variáveis IN e OUT, ambas do conjunto de cores (*colset*) State. Para uma referência completa sobre os conjuntos de cores e sintaxe do CPN ML, o leitor deve consultar [37].

2.6.2 Linguagem CPN ML

A linguagem *CPN ML* é uma extensão de uma linguagem de programação funcional chamada, *Standard ML* (SML) [38], desenvolvida na *Edinburg University*. *Standard ML* é uma

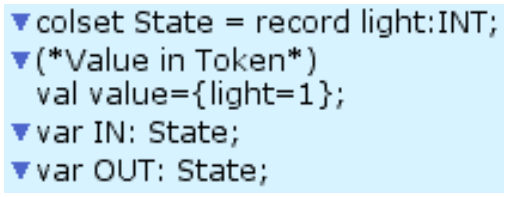


Figura 2.6: Declarações

linguagem de programação *Type-safe* que fornece um sistema de módulos ricamente expressivo e flexível para estruturar programas grandes, incluindo abstração, estrutura hierárquica e módulos genéricos.

Além disso, essa linguagem é facilmente adaptada para diferentes plataformas e outros tipos de implementações, porque tem uma definição precisa. O software *CPN Tools* [32] é um ambiente livre para o desenvolvimento de modelos CPN que adota a linguagem *CPN ML* para declarações e inscrições no modelo. A linguagem Standard ML foi escolhida como base da linguagem da ferramenta *CPN tools* devido à presença de algumas vantagens da linguagem, são elas:

(i) Uma linguagem consolidada e testada no ambiente de programação. Além disso, a criação de uma nova linguagem de programação é um processo muito lento e caro;

(ii) Ao invés de desenvolver um novo compilador a partir do início, era necessário apenas deixá-lo compatível com os sistemas operacionais e integrá-lo com o *CPN Tools*;

(iii) A grande quantidade de documentação e materiais disponíveis já existentes para o *Standard ML* e para linguagens funcionais;

(iv) O *Standard ML* possui funções de tipos, operações, variáveis e expressões, de maneira similar a uma linguagem funcional tipada. Por isso, foi conveniente a utilização desta linguagem;

(v) O *Standard ML* também tem uma sintaxe flexível e extensível para permitir que se escrevam as declarações e as inscrições de uma forma matemática.

O *Standard ML* possui compiladores disponíveis para uso comercial, e com essa linguagem é possível definir funções matemáticas contanto que sejam computáveis. Além disso, com essa

linguagem é possível decodificar funções e operações complexas. O *Standard ML* também torna possível realizar integrações entre segmentos de códigos. Um segmento de código é uma pequena parte de código sequencial anexado a uma transição que é executada cada vez que tal transição é disparada, e pode atualizar arquivos ou fazer outras formas de relatório, por exemplo.

Os modelos em CPNs abrangem três grupos: estruturais, declarações e inscrições [39]. As declarações e inscrições de modelos CPNs são realizadas através de extensões do Padrão ML (como *CPN ML*). Por outro lado, a estrutura de um modelo de CPN consiste basicamente em um gráfico marcado com lugares, arcos e transições. Nas linhas seguintes, são apresentadas algumas declarações e inscrições que adotam a linguagem *CPN ML*.

Declarações. Em CPNs, as declarações são usadas para definir conjuntos de cores, funções, variáveis e constantes.

Declarações do Conjunto de Cores. Dois exemplos de declarações de conjuntos de cores representados da seguinte forma:

colset REAL = real;

colset State = record light:int;

O primeiro exemplo mostra as declarações do conjunto de cores (*colset*), REAL, que considera valores reais. A segunda definição de conjunto de cores mostra uma maneira de representar tipos diferentes identificados pelo rótulo exclusivo, *state* (representa o estado do sistema em "0" ou "1"), que é representado pela variável *light* do tipo: *int*.

Declarações de variáveis. Uma variável é um identificador e seu valor pode ser alterado durante a execução do modelo. Os tipos das variáveis devem ser declarados em um dos conjuntos de cores definidos anteriormente, e essas variáveis podem ser usadas nas expressões de guarda, nos *"code segments"* - *action* - das transições, nas inscrições dos arcos e também nos lugares. Existem outros tipos de variáveis, chamadas de referência que podem ser utilizadas em todo o projeto. Essas variáveis de referência podem ser lidas e atualizadas por segmentos

de código. A palavra *var* foi adotada para declarar a variável e a palavra *globref* é a adotada para declarar variáveis de referência global, foi adotada desta forma por ser padronizada pelo CPNTools. Declarações de variáveis e uma variável de referência são exemplificadas de maneira em que *c* é uma variável do tipo *component* e *costEnergy* é uma variável de referência.

var c:*component*; (*Variável*) *globref costEnergy* = 0.11: real;(*Variável de referência*)

Declarações constantes. Nas declarações constantes, um valor é atrelado a um identificador que funciona como uma variável. Um exemplo é mostrado a seguir, no qual o *nMaxReplication* é o identificador e *val* a palavra adotada para declarar variáveis constantes, defnido pelo CPNTools.

val *nMaxReplication* = 100;

Declarações de função. Uma função pode ser considerada uma expressão que recebe parâmetros, ou seja, é uma expressão que precisa de argumentos antes de ser validada. Considere uma função soma, sendo dois números inteiros. A função *fun soma(x:int,y:int):int* recebe dois números inteiros e o retorna a soma. O tipo da função também é fornecido *int*, ou seja, uma função que recebe dois números inteiros e retorna um número inteiro [40]. As declarações, são escritas em CPN ML da seguinte forma:

- fun soma(x:int,y:int):int => soma(1,2);
val it = 3 : int
- fun soma(x:int,y:int):int => soma(2,2);
val it = 4 : int

Inscrições. As inscrições estão associadas aos componentes da CPN, como: lugar, arco e transição. Algumas inscrições podem ou não afetar o comportamento de uma rede.

Inserir Inscrições. Existem três inscrições que podem estar associadas a um lugar, sendo duas opcionais e uma é obrigatória:

- Inscrição do conjunto de cores (obrigatório): determina o tipo (cor) de todos os *tokens* que podem ser colocados no lugar.
- Inscrição de marcação inicial (opcional): especifica os *tokens* iniciais para o lugar.
- Inscrição do nome do lugar (opcional): identifica o lugar, podendo conter qualquer sequência de caracteres.

A Figura 2.7 mostra as inscrições que são associadas ao lugar 01. Nesta figura, o "01" corresponde à inscrição do nome do lugar; o *SB* é uma constante associada para especificar a marcação inicial; e *COMPONENT* é o conjunto de cores dos *tokens* que esse lugar pode receber.

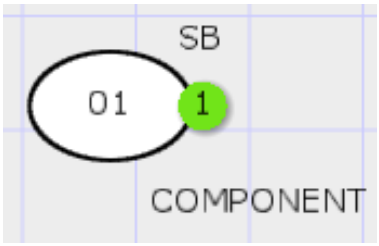


Figura 2.7: Inscrições do Lugar

Inscrições de Arco. Uma inscrição no arco é uma expressão *CPN ML* que avalia um conjunto múltiplo ou um único elemento, podendo também conter uma função. É importante afirmar que o conjunto de cores da expressão do arco deve corresponder ao conjunto de cores do lugar associado ao arco. A Figura 2.8 representa um arco com a inscrição *c* (variável do tipo *COMPONENT*).

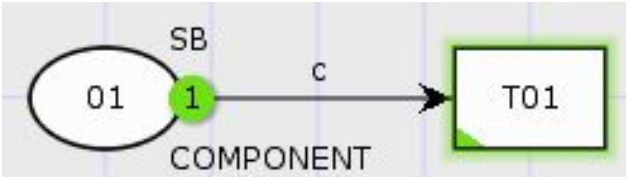


Figura 2.8: Inscrições do Arco

Inscrições da Transição. Existem quatro inscrições que podem estar associadas a transições, e todas elas são opcionais:

- Inscrição do nome da transição: um rótulo que identifica a transição e pode conter qualquer sequência de caracteres.
- Inscrição de guarda: uma guarda é uma expressão booleana de CPN ML que pode ser avaliada como verdadeira ou falsa.
- Inscrição de tempo: um atraso de transição pode ser associado de tal forma que se o horário atual for 10 e o tempo de atraso for "@ + 2", o registro de data e hora dos *tokens* enviados para os lugares de saída será "12". Uma inscrição de tempo ausente é equivalente a um atraso de zero.
- Inscrição do segmentação do código. Cada transição pode ter um segmento de código anexado que contenha um código ML. Segmentos de código são executados quando ocorre o disparo da transição.

A Figura 2.9 (a) é composta por 3 lugares e uma transição. A transição $T01$ está com a inscrição de guarda: $[x + y > 0]$ onde, esta transição somente estará habilitada se obedecer este critério. A inscrição de tempo está com *"@+1"*, ou seja, esta transição somente estará habilitada quando o tempo da simulação do sistema for igual a *1*. Por fim, a inscrição da segmentação do código está realizando a soma dos valores dos lugares *01* e *02*. A Figura 2.9 (b) mostra o resultado do disparo da transição *T01* quando todas as condições são obedecidas.

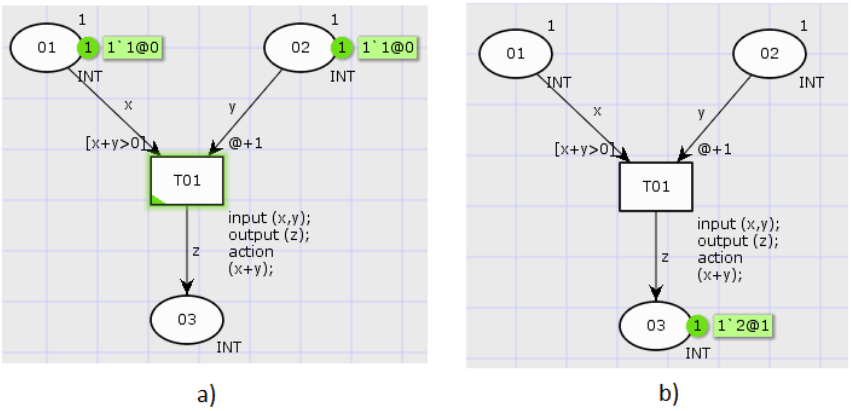


Figura 2.9: a) Inscrições da transição antes do disparo; b) Inscrições da transição depois do disparo

2.6.3 CPN Temporizado

Este trabalho adotou o modelo computacional TCPN (*Timed Coloured Petri Net*) [35], para garantir restrições de tempo. As subseções a seguir apresentam conceitos relacionados a este modelo. O TCPN considera um *clock* global que representa o tempo do modelo do sistema. Mais precisamente, cada *token* tem uma marcação de tempo. O estado atual de um modelo pode ser alterado quando o disparo da transição é habilitado. Em seguida, a transição é ativada quando o *clock* global for igual ao tempo associado ao *token*.

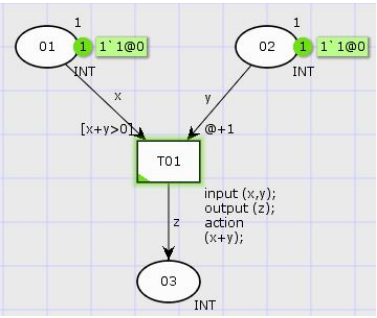


Figura 2.10: Modelo Temporizado

Para representar uma marcação de tempo do *token*, o operador "@" é adotado como padrão do CPNTools. A Figura 2.10 mostra um modelo que necessita do tempo para que a transição seja disparada. Neste modelo, existe uma condição adicional para a habilitação de uma transição que é relacionado com o tempo. Dessa forma, uma transição somente se encontrará habilitada quando o *clock* chega ao tempo associado aos *tokens* nos lugares de entrada da transição.

2.6.4 CPN Hierárquico

A ideia central de se trabalhar com CPN Hierárquico (HCPN) [31, 36] é permitir que o sistema seja construído com estruturas hierárquicas, representadas por transições de alto nível, chamadas de transições de substituição. Isso significa que é possível modelar grandes sistemas em CPNs e relacionar com CPNs menores, de maneira bem definida. Em determinado nível, é possível fornecer uma descrição simples da atividade modelada sem ter que considerar detalhes internos sobre como ela é executada. Em um nível mais baixo, é possível

especificar o comportamento detalhadamente. O modelo representado pela transição de substituição é denominado subpágina e o modelo superior, que possui a transição de substituição, é a página. Essas páginas são conectadas umas às outras pelos socket de entrada e pelos socket de saída, chamados de lugares de entrada e lugares de saída, respectivamente [32, 41].

2.7 Modelo de Fluxo de Energia (EFM)

O modelo de fluxo de energia, denominado EFM [13], representa o fluxo da energia elétrica que passa pelos componentes do sistema, considerando a sua eficiência e a capacidade máxima de potência que cada componente pode suportar. A Figura 2.11 apresenta, por exemplo, um modelo com 3 componentes de um *data center*: UPS, $SubPanel$ e $PowerStrip$.

O UPS é responsável por prover a energia de forma ininterrupta e estabilizada, $SubPanel$ é o quadro de energia do sistema e $PowerStrip$ é uma régua de alimentação de energia elétrica. Cada componente possui atributos como: eficiência, custo de aquisição, e a capacidade máxima de potência que o equipamento consegue suportar.

Figura 2.11: Modelo de Fluxo de Energia

Modelos EFM's são responsáveis por computar também a energia consumida pela arquitetura, o custo e a eficiência energética do sistema modelado. Esses resultados auxiliam projetistas de *data centers* a propor um novo ambiente ou otimizar um já existente. Dessa forma, futuras arquiteturas podem ser projetadas impactando menos o meio ambiente e com uma maior disponibilidade.

Fluxo de Energia nos Equipamentos. A verificação do fluxo de energia que passa pelos equipamentos é calculada para que não exceda a capacidade máxima de potência que cada equipamento pode prover.

No modelo EFM, quando a energia excede a potência do equipamento o sistema se torna falho do ponto de vista energético (mesmo podendo estar funcional em relação à disponibilidade de todo o sistema).

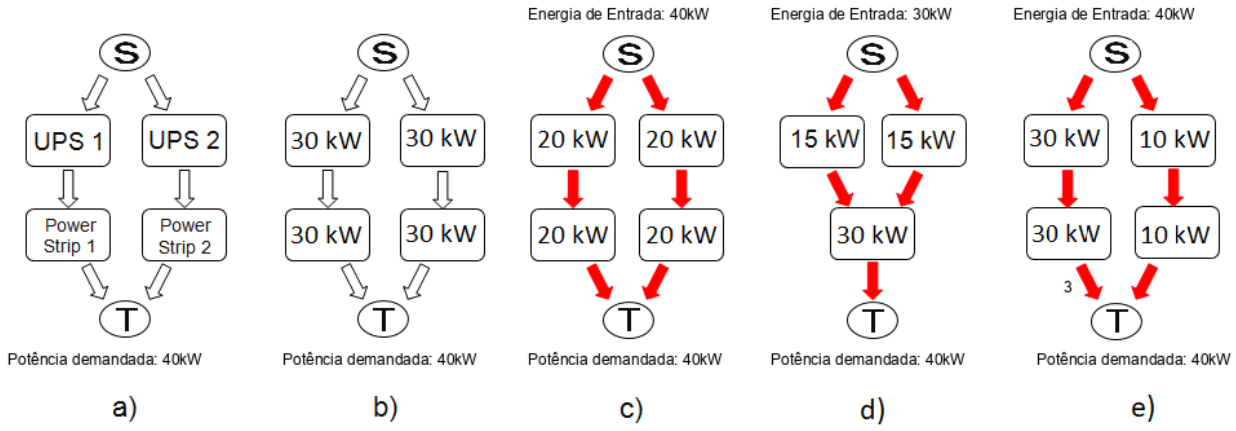


Figura 2.12: a) Exemplo de sistema; b) Capacidade Máxima de energia; c) Fluxo de energia com Sucesso; d) Fluxo de energia com falha; d) Representação com peso nos arcos.

A Figura 2.12 (a) mostra uma infraestrutura de *data center* contendo 4 equipamentos. Assumindo que a energia demandada pelo sistema de *data center* corresponde a $40kW$ (valor associado ao nó T) e a capacidade máxima de energia que o equipamento pode suportar, são representados na Figura 2.12 (b). A Figura 2.12 (c) mostra um possível fluxo de energia, no qual a energia fornecida por cada UPS é de $20kW$, que é transferida para os $PowerStrips$ ($20kW$).

Em outro exemplo, ao invés de adotar dois $PowerStrips$, será considerado apenas um. Este sistema está representado na Figura 2.12 (d) que é possível suportar $30kW$ de potência demandada (valor associado ao nó T). Assim, o sistema não é capaz de suportar a demanda de energia. A Figura 2.12 (e) mostra um sistema com duas réguas de energia($PowerStrip1$ e $PowerStrip2$). O $PowerStrip1$ é capaz de fornecer três vezes mais energia do que o $PowerStrip2$. Dessa forma, o comportamento do sistema é especificado com os pesos associados nas arestas do gráfico.

Se a energia fluindo em qualquer um dos equipamentos superar a capacidade máxima que o equipamento pode suportar, significa que existe uma sobrecarga e o sistema será

considerado falho do ponto de vista energético.

Quantificando o custo. O modelo EFM considera o custo de aquisição dos equipamentos dos *data centers* e os custos operacionais. O custo de aquisição (CA) corresponde aos recursos necessários para implementar a infraestrutura do *data centers*. O custo operacional (CO) é o custo para manter o sistema no modo operacional, que é representado pela seguinte equação:

$$CO = E_{entrada} \times T \times E_{custo} \times D \tag{2.6}$$

onde $E_{entrada}$ é a energia elétrica consumida; T é o período considerado; E_{custo} corresponde ao preço da energia; e D é a disponibilidade.

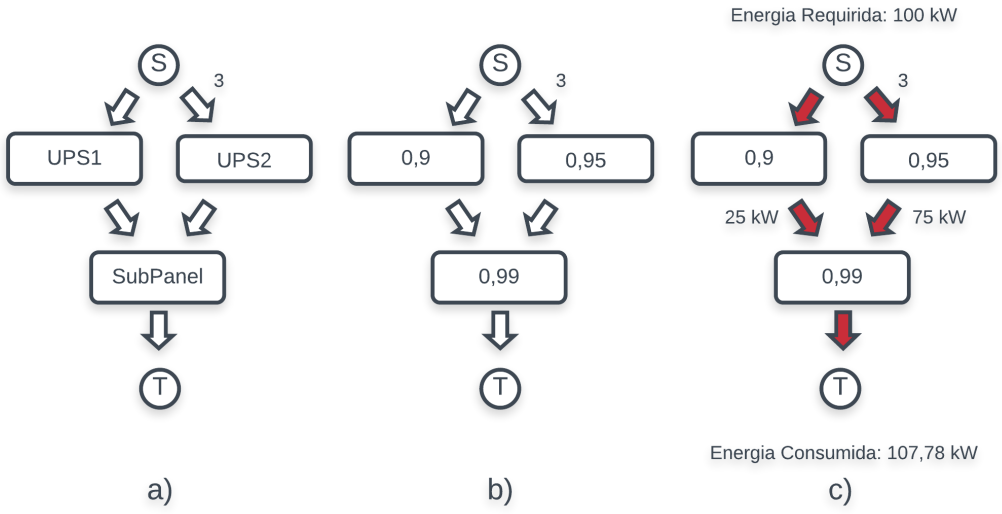


Figura 2.13: a) Sistema Simples; b) Valores da Eficiência; c) Energia Consumida.

A Figura 2.13 (a) representa um sistema simples de energia elétrica, composto por dois UPS e um powerstrip, que fornece 100 kW de energia para os dispositivos de TI. Neste exemplo, os pesos atribuídos nas arestas que fazem a conexão com o equipamento UPS1 e UPS2 ao componente Powerstrip são, respectivamente, um e três. Desta forma, o componente UPS2 fornece três vezes mais energia que o UPS1. As eficiências de UPS1 e UPS2 o Powerstrip são de 0,9, 0,95 e 0,99 (ver Figura 2.13 (b)), respectivamente. Para fornecer a

potência exigida de 100 kW, a energia consumida é de 107,78 kW, como mostra a Figura 2.13 (c).

Eficiência da Arquitetura. É necessário conhecer a eficiência dos equipamentos, pois se o equipamento tiver uma eficiência baixa, maior será o consumo de energia deste equipamento. A verificação da eficiência da arquitetura é representada pela seguinte equação:

$$Ef = \frac{E_{entrada}}{E_{saída}} \tag{2.7}$$

onde Ef é a eficiência da arquitetura, $E_{entrada}$ é a energia demandada pelo sistema de TI do *data centers*, e o $E_{saída}$ é a energia de fato consumida decorrente da eficiência energética dos dispositivos que acabam por ter energia dissipada, por exemplo, em forma de calor.

Downtime. É o tempo em que o sistema ficou indisponível durante um período de tempo T. Essa indisponibilidade pode ser decorrente de algum problema específico ou alguma manutenção programada. O *downtime* é representado pela seguinte equação:

$$downtime = (1 - D) * T \tag{2.8}$$

onde D é a disponibilidade da arquitetura e T o período de tempo.

Quantificando o consumo da exergia operacional. A exergia operacional é utilizada para identificar o sistema elétrico com maior eficiência energética. A Equação que calcula o consumo da exergia operacional é:

$$Ex_{op} = \sum_{i=1}^{n} Ex_{op_i} \times T \times D \tag{2.9}$$

onde Ex_{opi} é a exergia operacional de cada equipamento, T é o tempo em que o sistema estará em funcionamento, e D é a disponibilidade do sistema.

Já para o cálculo da exergia de cada dispositivo elétrico é mostrado na Equação 2.10:

$$P_{entrada} \times (1 - \eta) \tag{2.10}$$

onde η é a eficiência energética do dispositivo e o $P_{entrada}$ é a potência de entrada no equipamento.

2.8 Resumo

Neste capítulo foram descritos os principais conceitos necessários para a compreensão desta dissertação. Primeiramente, foi realizada uma visão geral do processo de simulação. Depois disso, conceitos sobre *data centers* e toda sua infraestrutura, seguindo por uma visão geral sobre disponibilidade e exergia. Posteriormente, foram apresentadas as redes de Petri elementares e as coloridas e, por fim, o modelo EFM foi apresentado.

Capítulo 3

Trabalhos Relacionados

Este capítulo mostra trabalhos relacionados à pesquisa desenvolvida. No início, são apresentados trabalhos relacionados ao consumo energético e, em seguida, são apresentados trabalhos relacionados a confiabilidade.

3.1 Consumo energético

Callou et al. [42] apresentam uma abordagem baseada em redes de Petri coloridas (CPN) para estimar o tempo de execução e o consumo de energia de sistemas embarcados. Nos modelos CPN propostos foram utilizados códigos probabilísticos para representar o fluxo de código de sistemas embarcados e, assim, computar o tempo de execução e verificar o consumo de energia decorrente de sua execução. Este artigo teve o foco restrito aos sistemas embarcados.

Rocha et al. [43] apresentam a relação entre a infraestrutura elétrica e a disponibilidade de softwares em um *data center*. Para isso, foram utilizados modelos em SPN para representar o comportamento da energia, e o RBD para estimar a disponibilidade. Não foi o foco do trabalho a quantificação do impacto da disponibilidade no comportamento do modelo de fluxo de energia.

Wang et al. [44] apresentam uma modelagem de um sistema de manufatura para avaliar o

consumo de energia em máquinas industriais. Modelos em CTPNs (*colored timed petri nets*) foram utilizados para estimar o consumo de energia das máquinas. Nesta pesquisa, foi verificado que métodos matemáticos estocásticos tradicionais são insuficientes para tal objetivo e, dessa forma, optou-se pelas CTPNs. Para estudar o desempenho dinâmico dos modelos, é necessário considerar o tempo de duração das atividades, portanto, foram atribuídos tempos de disparo às transições.

Callou et al. [12] sugerem um conjunto de modelos para à quantificação do impacto da sustentabilidade, custo e confiabilidade das infraestruturas de resfriamento do *data center*. Além disso, este modelo verifica a capacidade máxima que o componente pode suportar. Para isso foi utilizado o SPN e RBD, para realizar as estimativas. Já para o fluxo de energia, foi utilizado o EFM. A presente pesquisa se baseou no conceito do EFM para a verificação do fluxo energético do *data center*.

3.2 Confiabilidade

Andrade et al. [45] propõem um conjunto de modelos em SPN para quantificação da disponibilidade e da segurança contra catástrofes de *data center*. O trabalho atual, além de calcular a disponibilidade da arquitetura, verifica o consumo de energia dos equipamentos quantificando o custo da arquitetura.

Talebberrouane et al. [46] apresentam dois modelos diferentes para analisar cenários de disponibilidade e indisponibilidade de sistemas críticos de segurança. Os modelos utilizados foram em redes de Petri estocásticas generalizadas (GSPN), cadeias de Markov e árvore de falhas. Não foi o foco da pesquisa o consumo de energia.

Melo et al. [47] propõem um modelo para avaliar ambientes de computação em nuvem, levando em consideração a quantidade de recursos disponíveis. Neste trabalho foi utilizado o RBD para a estimativa da sua disponibilidade, como também SPN. O presente trabalho propõe a utilização de modelos em CPN para estimar a disponibilidade, como também o fluxo energético.

Sousa et al. [48] propõem uma estratégia para modelagem de infraestrutura de nuvem

hierárquica e heterogênea, para descrever hierarquicamente as dependências entre componentes e a infraestrutura de nuvem. Foi utilizado para estimar a dependabilidade (disponibilidade, confiabilidade e o *downtime*) a técnica de modelagem RBD e para o comportamento o SPN. Além disso, esta abordagem permite a seleção de infraestruturas em nuvem de acordo com essa avaliação da confiabilidade e dos requisitos de custo.

Nguyen et al. [49] apresentam a importância de ter uma infraestrutura de *data center* tolerante a falhas e que permaneça com suas operações contínuas de processamento de dados. Dessa forma, irá garantir a continuidade operacional e o prolongamento do mais alto nível de confiabilidade/disponibilidade dos componentes e de seus subsistemas físicos. Para avaliar a confiabilidade (confiabilidade, disponibilidade, performabilidade, etc.) de um determinado sistema, o uso de modelos matemáticos, que normalmente inclui modelos de espaço de estados e modelos hierárquicos, são utilizados, como exemplo: cadeias de Markov de tempo contínuo (CTMC) e redes de Petri estocásticas (SPN).

Wiboonrat [50] apresenta um diagnóstico de avaliação de confiabilidade para *data centers*. Foi utilizado o FMECA (Modelos de falha, efeitos e análise de criticidade) e o RBD para realizar a estimativa.

Liu et al. [51] propõem uma modelagem de um sistema de requisições de pedidos de determinado serviço em um *data center*, de forma integrada, onde é analisado quando o serviço está inoperante e o tempo de espera pela requisição do serviço solicitado. Para isso, é utilizando o modelo CGSPN (*colored generalized stochastic Petri net*). O presente trabalho, além de analisar a disponibilidade do serviço de forma dinâmica, analisa o consumo energético do sistema modelado.

Pinna et al. [52] propõem um estudo sobre o IDPDA (*Integrated Deterministic and Probabilistic Dependability Analysis*) e acentua em um sistema redundante e reconfigurável, e poderá compensar eventuais falhas. Para isso é utilizado a técnica de modelagem CPN. Neste trabalho, o estudo de caso apresenta, turbopropulsores paralelos de água de alimentação, onde se um desses equipamentos falhar, o outro caminho assume o fluxo.

Song [53] propõe a construção de modelos para a manutenção preventiva e corretiva de sistemas ferroviários, considerando confiabilidade e disponibilidade. No estudo de caso deste

trabalho são verificados sistemas em série e sistemas em paralelo, utilizando a técnica de modelagem CPN que envolve tanto a lógica de falha quanto os estados do sistema para executar as estratégias corretivas e de manutenção periódica.

Wang et al. [54] sugerem modelar um sistema com multi-estados, envolvendo diversos componentes que estão sujeitos ao processo de degradação, como também, aos processos de manutenção corretiva e manutenção preventiva. Com a complexidade do sistema, devido a sua estrutura de dependências entre suas unidades, foi adotada a técnica de modelagem CPN que considera unidades de sistemas com degradações [55].

Fitch et al. [56] propõem um mecanismo de segurança para armazenamento de informações baseado em nuvem segura e tolerante a falhas. A técnica de modelagem utilizada foi CPN, neste modelo é utilizado diversos provedores de nuvem como um *cluster* para prover tal serviço. Nesta abordagem, é suportado a manutenção da confidencialidade dos dados armazenados, como também garante que a falha ou o comprometimento de um provedor de nuvem individual em um *cluster* de nuvem não resultará em um comprometimento total dos dados. O exemplo de estudo de caso utilizado foi um sistema de armazenamento online de registros médicos.

A Tabela 3.1 representa um resumo que compara os principais assuntos focados neste trabalho com os trabalhos mencionados neste capítulo. Pode-se concluir que nenhum trabalho faz esse estudo quantificando o impacto da disponibilidade e sustentabilidade no fluxo energético em tempo de execução. Este trabalho se difere porque propõe quantificar o impacto da disponibilidade no fluxo energético em tempo de execução.

3.3 Resumo

Este capítulo apresentou os principais trabalhos relacionados com sustentabilidade e confiabilidade. O presente trabalho computa em apenas uma técnica de modelagem às métricas desejadas e, principalmente, verifica o impacto da disponibilidade dos equipamentos que compõe o sistema de potência de data centers, no fluxo energético.

Trabalhos	Consumo energético	Custo	Disponibilidade	Modelagem
[42]	x			CPN
[43]	x		x	RBD
[44]	x			CTPN
[12]	x	x	x	GSPN/RBD
[45]			x	SPN
[46]			x	GSPN
[47]			x	RBD
[48]		x	x	RBD
[49]			x	CTMC
[50]			x	RBD
[51]			x	CGSPN
[52]			x	CPN
[53]			x	CPN
[54]			x	CPN
[56]			x	CPN
Este trabalho	x	x	x	CPN

Tabela 3.1: Resumo dos trabalhos relacionados.

Capítulo 4

Metodologia

Este capítulo descreve a metodologia proposta para o entendimento e a construção dos modelos propostos neste trabalho. Em seguida, é apresentada uma proposta para a criação do modelo CPN através de um software desenvolvido pelo grupo de pesquisa GMOS (Grupo de Modelagem e Otimização de Sistemas) do programa de pós graduação em informática aplicada. Dessa forma, o usuário não precisa de um conhecimento específico em CPN para a sua criação.

4.1 Metodologia

Esta seção apresenta a metodologia adotada (ver Figura 4.1) para fazer a verificação do fluxo de energia e computar a disponibilidade, custo e impacto ambiental de sistemas computacionais como os *data centers*. O primeiro passo da metodologia é a compreensão do sistema. Neste momento é fundamental identificar os componentes que integram o *data centers*, e como eles estão conectados. É necessário conhecer a organização entre os componentes que compõem o sistema em análise para a modelagem da arquitetura.

O segundo passo consiste na criação dos modelos em CPN. Para isso, adotou-se a estratégia da representação de apenas um modelo, para computar os valores de disponibilidade e o de fluxo de energia.

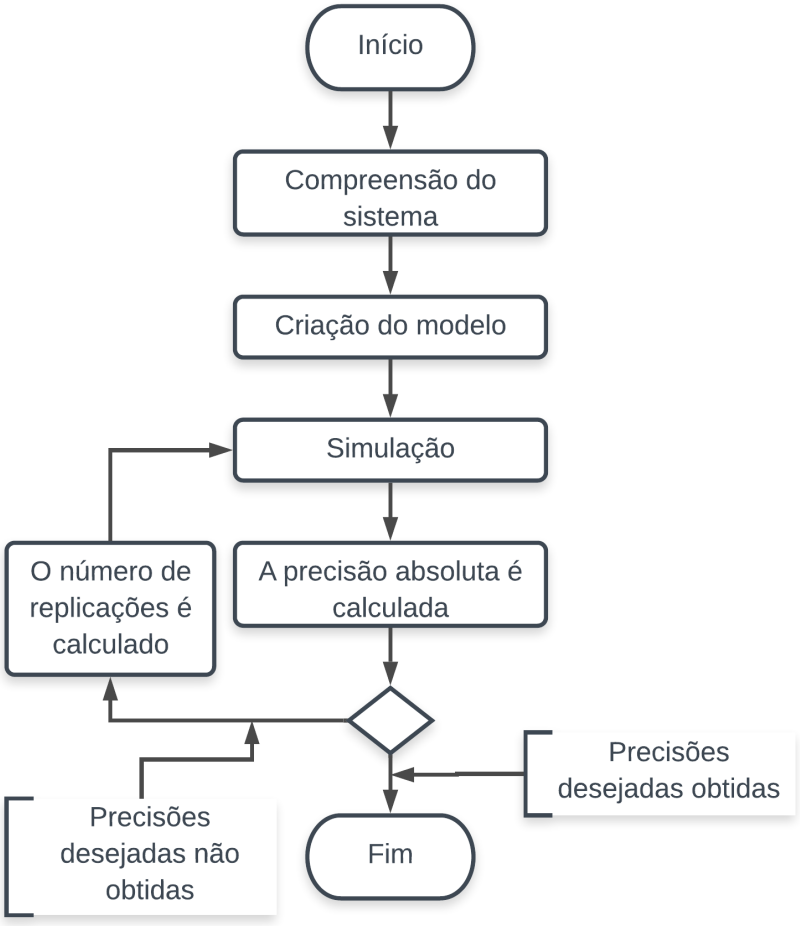


Figura 4.1: Fluxo da metodologia

Na terceira etapa (simulação), o modelo tem como valor de entrada a energia demandada pelo sistema de TI dos *data centers*, o tempo de análise do sistema (para computar o custo operacional) e o valor do kWh da energia elétrica. A disponibilidade da arquitetura é calculada na simulação do modelo. Os valores de entrada para o cálculo da disponibilidade são representados pelo MTTF e MTTR de cada equipamento.

A execução do modelo somente finaliza quando se é atingido o critério de parada. A avaliação de critério de parada adotada, foi escolhida para fornecer os resultados de simulação levando em consideração o grau de confiança e o erro especificado [57].

O critério de parada adotado considera as precisões absolutas para o consumo de energia, médias e desvios padrão. A precisão absoluta é calculada após a execução de 10 replicações

do experimento conforme mostra a Equação 4.1.

$$PrecisaoAbsoluta = t_{1-\alpha/2,n-1} \times \frac{s}{\sqrt{n}} \tag{4.1}$$

onde: t é o valor calculado para *1-*$\alpha/2$ graus de confiança e *n-1* graus de liberdade; s é o desvio padrão e n o número de replicações no exemplo.

As precisões computadas para o consumo de energia são comparadas com os erros desejados. A simulação é concluída se esses valores calculados forem menores que a precisão desejada, caso contrário, a simulação prossegue calculando o número necessário de replicações através da Equação 4.2. Existe apenas um valor de replicação, o consumo de energia elétrica.

$$i = \left[\frac{t_{1-\alpha/2,n-1} \times s}{PrecisaoDesejada}\right]^2 \tag{4.2}$$

Se as precisões desejadas foram alcançadas, o sistema exibe os resultados. Caso contrário, o fluxo de execução segue. Após a simulação, obtêm-se as métricas de interesse como o consumo de energia, a eficiência energética da arquitetura e a disponibilidade.

4.2 Protótipo

A ferramenta em desenvolvimento tem como objetivo o de auxiliar os projetistas (i.e., a computar métricas de disponibilidade, consumo de energia, sustentabilidade e custo) sem precisar de um conhecimento prévio do formalismo empregado. A Figura 4.2 mostra a tela inicial da ferramenta que faz a conversão automática desse modelo representado para o formalismo de CPN. Nessa ferramenta, um Parser é o responsável por converter o modelo da ferramenta proposta para modelos formais (SPN, RBD e CPN). Neste trabalho, utilizamos a ferramenta para converter uma representação de alto nível (ver Figura 4.2) para o padrão reconhecido pelo CPN Tools. Sendo assim, o Parser se comunica com o CPN Tools para permitir a avaliação e análise de modelos em CPN.

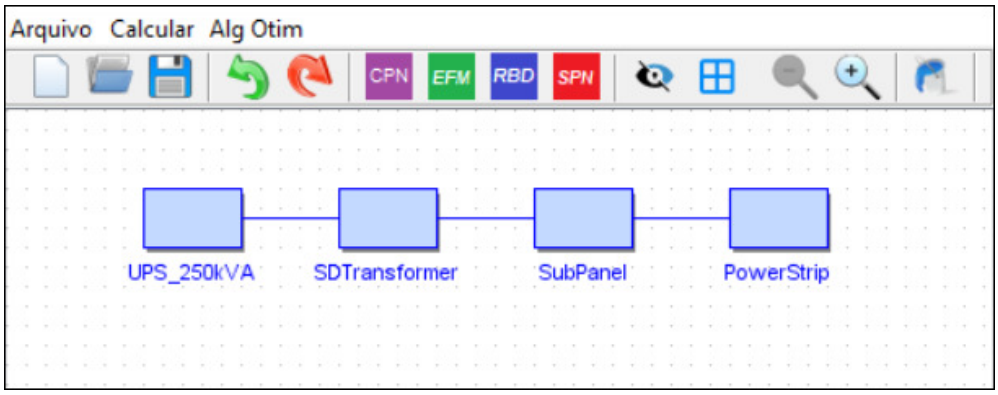


Figura 4.2: Ferramenta proposta

Metodologia do protótipo. Neste momento será apresentada a metodologia utilizada para o desenvolvimento desta ferramenta. A Figura 4.3 mostra a integração da ferramenta com ambientes computacionais já existentes. Pode-se observar que além da ferramenta desenvolvida com a visão única da representação dos sistemas computacionais de interesse, foi proposto também um Parser que é o responsável por converter o modelo da ferramenta proposta para modelos formais. Essa conversão será feita de acordo com o desejo do projetista, onde o Parser terá que realizar a conversão do modelo proposto para modelos formais (SPN, RBD e CPN).

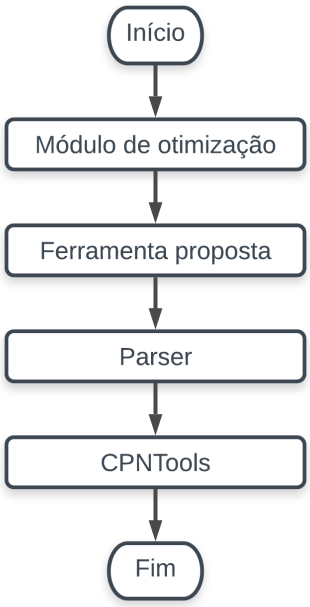


Figura 4.3: Metodologia do Software

É importante mencionar a aplicabilidade do ambiente proposto, tendo em vista que a partir dessa ferramenta será possível desenvolver métodos de otimização que fazem uso de técnicas de modelagem distintas. Dessa forma, técnicas multiobjetivo de otimização que façam uso de forma integrada dos formalismos de redes de Petri coloridas (CPN), redes de Petri estocásticas (SPN), modelo de fluxo de energia (EFM), por exemplo, possam ser implementadas de forma automatizadas com a ferramenta proposta.

Etapas do processo de conversão. A conversão de alto nível (software proposto) para os modelos formais acontece da seguinte forma: dentro de cada retângulo (vértice) são armazenados os dados básicos (MTTF e MTTR) utilizados pelos formalismos RBD, SPN e CPN para o cálculo de disponibilidade, por exemplo. Internamente na ferramenta existe um Parser para cada um dos formalismos, onde são executados os seguintes passos: (i) os vértices são convertidos para a representação de um equipamento do formalismo associado ao Parser; (ii) é realizada uma filtragem para se identificar os vértices iniciais; (iii) esses vértices passam por um processo de mapeamento dos caminhos até os vértices finais, realizado através de uma adaptação do algorítimo de busca em profundidade; (iv) por último, esses caminhos são convertidos para a representação do formalismo desejado.

Capítulo 5

Modelo

Este capítulo apresenta os modelos, em redes de Petri coloridas, que serão utilizados no estudo de caso. O capítulo inicia com a apresentação de um modelo básico com apenas um componente, detalhando a sua estratégia hierárquica adotada pelo modelo CPN, vale destacar que no modelo básico o objetivo é mostrar a estrutura do sistema que será utilizado. No exemplo posterior, serão consideradas as demais explicações (ex. funções, cálculos etc.). Posteriormente, é apresentado um modelo em série mostrando todas as funções de cálculo do sistema e, por fim, um modelo redundante mostrando como é feita a escolha do fluxo energético quando um dos caminhos apresenta a inoperância. O modelo proposto recebe como parâmetros: custo de aquisição, eficiência energética, MTTF e MTTR para cada equipamento presente no modelo, como também a disponibilidade de toda a arquitetura. Dessa forma, é possível utilizá-los para calcular as métricas equivalentes para o sistema como um todo. Vale destacar que quando a arquitetura possui mais de um caminho para a transmissão da energia elétrica, o modelo proposto consegue identificar antecipadamente se algum destes caminhos encontra-se inoperante, e redireciona o fluxo energético para o caminho operante.

5.1 Modelo básico

Nesta sessão será apresentado um modelo básico em rede de Petri colorida para uma arquitetura elétrica básica composta por apenas um componente. A Figura 5.1 ilustra o sistema a ser modelado.

Figura 5.1: Sistema básico com 1 componente

A Figura 5.2 mostra o modelo correspondente em rede de Petri colorida para um componente (ver Figura 5.1). Este modelo é composto por 4 lugares e 2 transições. Um *token ini* (do tipo $REQUIREMENT\_LIST$ ver Figura 5.3) no lugar 01 representa o início do modelo. Esse *token* carrega todos os atributos específicos que o modelo de fluxo de energia (EFM) necessita para a sua representação e cálculo das métricas (ex., eficiência energética, custo, etc). Os lugares ON e OFF representam os estados ligado e desligado do equipamento. No lugar ON existe um *token equi* (do tipo $COMPONENT$ ver Figura 5.4) que representa um componente da rede elétrica. A transição T00 (através de sua subpágina) mostra a estrutura interna dos diversos cálculos das métricas de interesse do sistema.

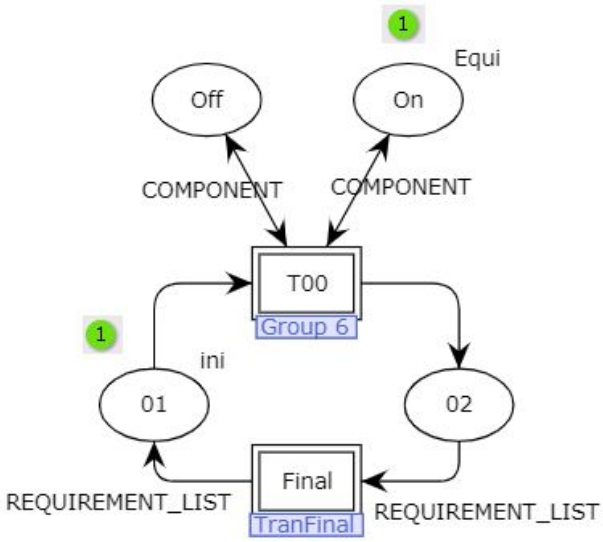


Figura 5.2: Modelo básico com 1 componente em CPN.

Vale ressaltar que as transições T00 e Final têm associado um submodelo (ou subpágina),

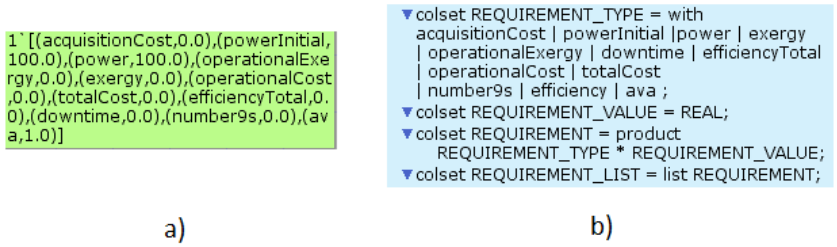


Figura 5.3: *token (ini)*

ou seja, existe uma estrutura interna para cada transição do modelo da Figura 5.2. Como cada transição de substituição representa uma instância, se for criada uma outra transição (ou instância) para representar um componente elétrico, essa transição também terá a mesma estrutura (ver Figura 5.5). Os atributos específicos que cada componente da rede elétrica carrega estão presentes no *token* do tipo $COMPONENT$.

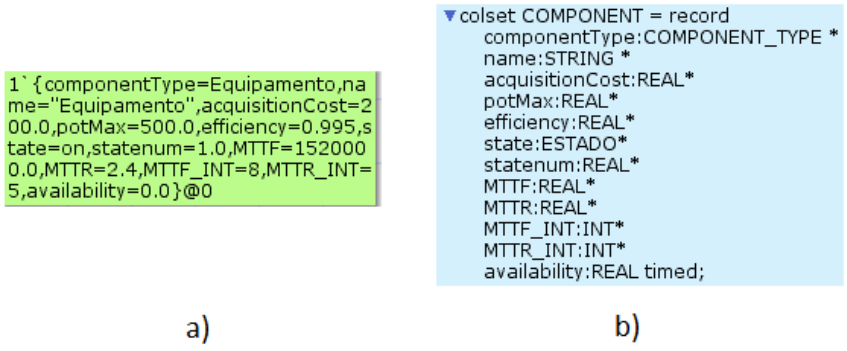


Figura 5.4: *Token* representando o equipamento/componente

As regras para implementar grupos de transições (ex., T00) nos modelos CPN, permitem transformar uma superpágina de um modelo CPN (modelo de mais alto nível) em uma subpágina do modelo para simplificar a representação do modelo na visão de mais alto nível. Todas as características para estimar os valores do fluxo de energia em conjunto com as verificações de capacidade máxima de potência dos equipamentos, e para se computar as métricas de disponibilidade, custo, eficiência, consumo de energia, etc, são preservadas.

A Figura 5.5 representa a transição T00 do modelo CPN básico. É importante salientar que as funções que computam os valores do fluxo de energia e da disponibilidade são representadas no modelo hierárquico na subpágina da transição T00. Assim, quando a transição T00 ocorre, os valores do fluxo de energia e da disponibilidade relacionados aos equipamentos

são computados.

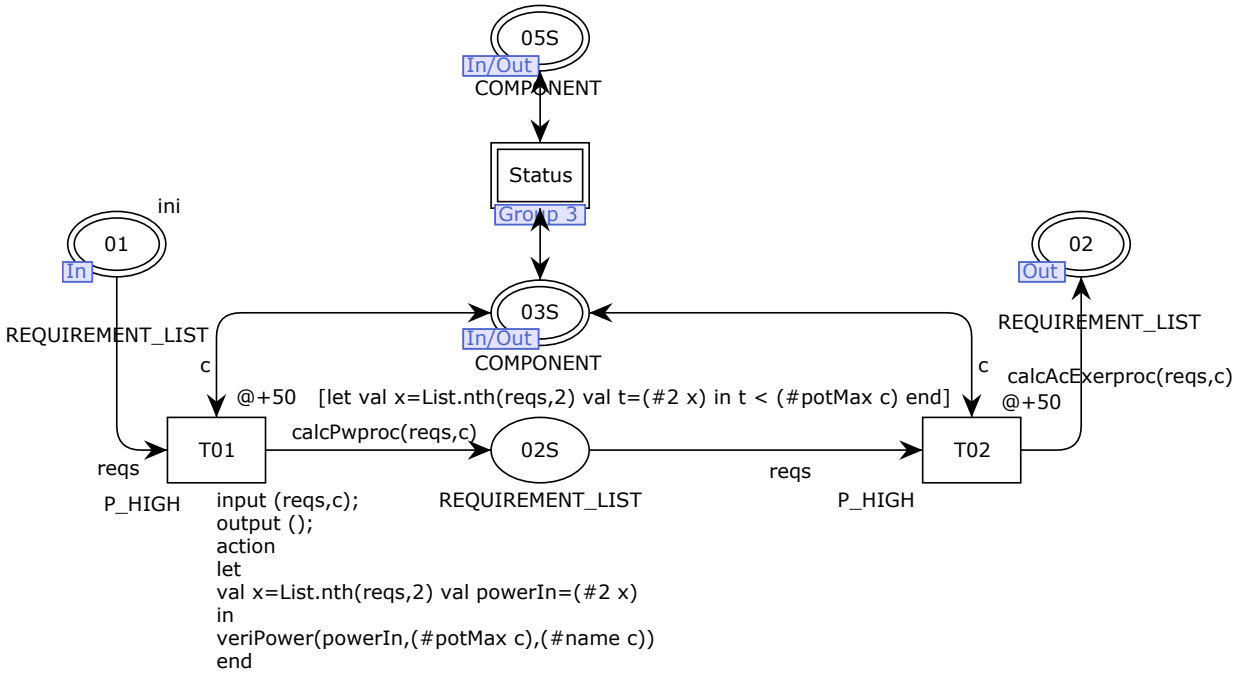


Figura 5.5: Modelo CPN subpágina (equipamentos).

A Figura 5.6 é a subpágina representada pela transição Status presente na Figura 5.5. Esta transição é responsável por computar o estado do componente, como também a disponibilidade do mesmo. Esta transição somente estará habilitada quando for respeitado a sua função de guarda, ou seja, a transição T03 será habilitada quando o tempo @+($\#MTTF\_INTc$) estiver chegado, em outras palavras, quando o tempo global gerenciado pela simulação, atingir o valor presente na expressão *@+(#MTTF_INT c)*, a transição estará habilitada para o disparo. A letra *c* nesta expressão, representa o valor do MTTF associado ao *token* do tipo *COMPONENT*. Da mesma forma, a transição T04 com o tempo de guarda *@+(#MTTR_INT c)* só estará habilitada quando o tempo simulado pela ferramenta estiver no valor correspondente ao associado a transição (valor MTTR_INT presente no token *Equi* da Figura 5.2). A transição T03 (ver Figura 5.6) representa a falha do equipamento, e o T04 representa o reparo.

A Figura 5.7 representa a subpágina da transição Final (ver Figura 5.2). Nesta subpágina existem 4 lugares e 3 transições. Um *token* no lugar 01F habilita a transição TF03, e o disparo desta transição inicia o fluxo desta subpágina. As funções presentes neste modelo

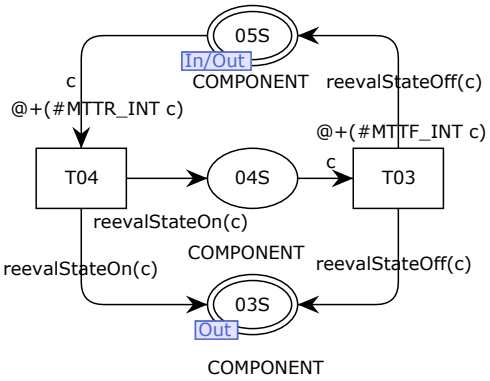
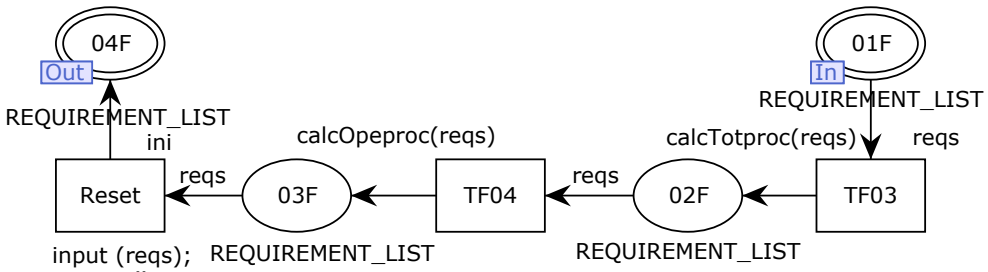


Figura 5.6: Transição Status

serão detalhadas na próxima seção. Após o disparo da transição T04, computam-se as operações finais do sistema, como por exemplo, o custo operacional do modelo. Por fim, o disparo da transição Reset representa o início de uma nova execução (caso o critério de parada não tenha sido obtido) ou representa o término da simulação do sistema.




```
output ();
action
let val powerEnd = List.nth(reqs,2);
val powerEndd = (#2 powerEnd);
val meanPower=average(!meanEnergyList);
val powerError=standardError(!meanEnergyList);
val powerSD=standardDeviation(!meanEnergyList);
in
addIterData(powerEndd) ;
if checkEnd() then(
  isEnd:=1;
  writeFile1("\n\nMedia Global Da Energia: " ^ Real.toString meanPower^ "\n" ^
             "Desvio Padro da Energia: " ^ Real.toString powerSD ^ "\n" ^
             "Erro da Energia " ^ Real.toString powerError ^ "\n" ^
             "\n")
)
else
   newIter();
  isEnd:=0
end;
```

Figura 5.7: Transição TranFinal

5.2 Modelo em série

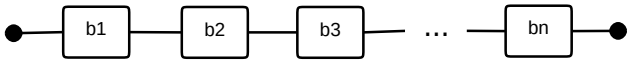


Figura 5.8: Sistema em série.

Na disposição em série, o sistema somente funcionará se todos os seus componentes estiverem ativos. A Figura 5.8 exemplifica este sistema. Neste exemplo, considera-se o sistema mais simples, sem replicação. Vale ressaltar que a disponibilidade da arquitetura é simulada em tempo de execução e seu impacto no modelo de fluxo de energia é quantificado. A Figura 5.9 (a) demonstra um sistema simples de *data center* composto por 4 equipamentos, são eles: PowerStrip, SubPanel, SDT e UPS1. Neste modelo, o fluxo energético passará apenas por um caminho. A Figura 5.9 (b) mostra o fluxo energético quando não existe falha elétrica em nenhum dos caminhos com a energia requerida de 100 kW, chegará no final do sistema com a energia de fato consumida de 107,60 kW. Já a Figura 5.9 (c) mostra o fluxo energético com a ocorrência de uma falha elétrica no equipamento SDT. Como a energia não tem outro caminho para prosseguir, o sistema do ponto de vista elétrico também estará falho.

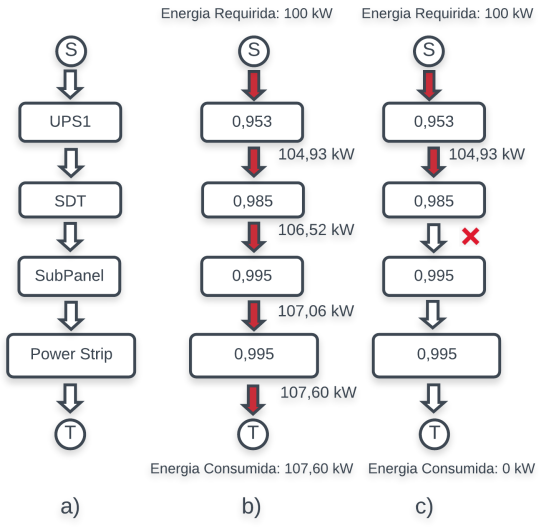


Figura 5.9: a) Sistema Simples; b) Sistema operante; c) Falha do UPS1.

A Figura 5.10 mostra a arquitetura de *data center* em série, com todos os seus equipa-

mentos, que será usada para demostrar a aplicabilidade do formalismo proposto. A Figura 5.11 apresenta o modelo CPN proposto, referente a arquitetura A1 ilustrada na Figura 5.10. A ferramenta adotada para modelagem e simulação dos modelos propostos foi o CPNTools [32].

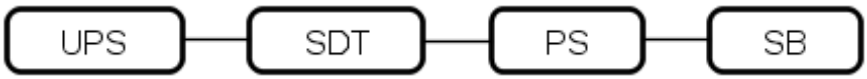


Figura 5.10: Arquitetura A1 (em série).

Esta arquitetura tem os seguintes equipamentos: *Power Strip, Sub Panel, SDT e UPS*, representados no modelo *CPN* pelas transições PS, SB, SDT, UPS. Todas essas transições (PS, SB, SDT, UPS) deste modelo CPN possuem subpáginas, ou seja, possuem um outro modelo de mais baixo nível e que é representado na visão de mais alto nível pela transição. Vale destacar ainda que todas essas transições são instâncias da mesma transição, e, todas as configurações que estão em uma transição são replicadas nas outras subpáginas. A construção do modelo dessa forma fornece mais consistência, onde cada componente elétrico do *data center* faz uso dos mesmos modelos, funções e variáveis pré-definidas no modelo CPN.

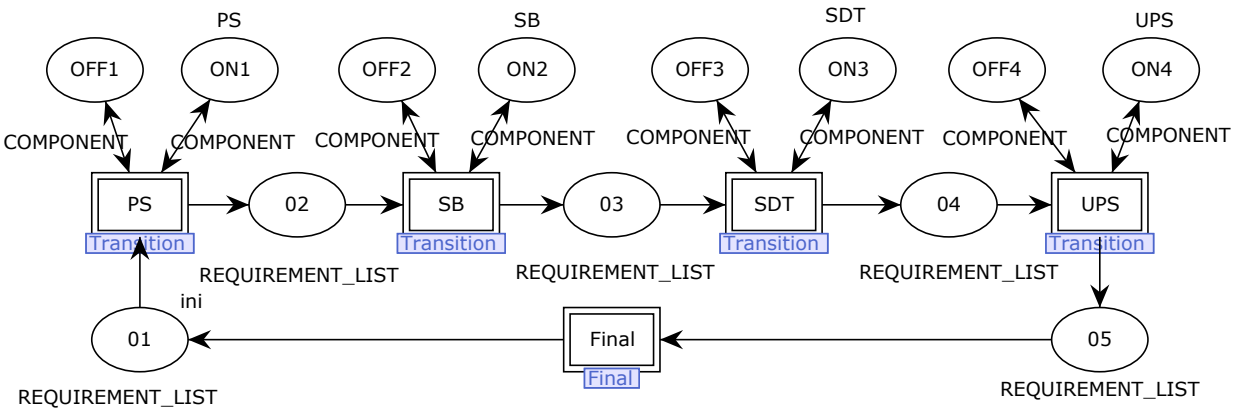


Figura 5.11: Modelo *CPN* - Arquitetura A1

Os atributos dos *tokens* (do tipo *COMPONENT*) presentes nos lugares ON ou OFF (ver Figura 5.11) são de acordo com os valores setados como parâmetros para o MTTF, MTTR e custo, por exemplo. Sendo assim, durante a simulação o *token* carrega os atributos de cada equipamento. Para modificar a estrutura para uma arquitetura em paralelo é necessário adicionar um caminho extra e, assim, se distribuir a energia por mais de um caminho acessível.

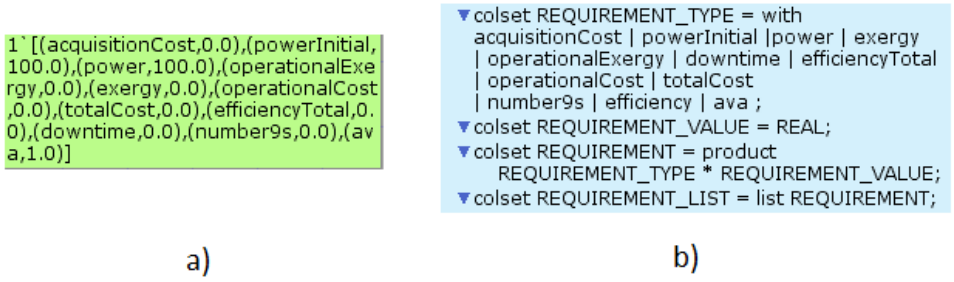


Figura 5.12: *token (ini)*

O *token ini* no lugar 01 (ver Figura 5.11) habilita a transição PS que representa o início da execução do modelo. O *token ini*, mostrado na Figura 5.12 (a), carrega todos os atributos que serão utilizados durante a simulação do modelo. A variável *powerInitial* representa o valor da energia demandada do sistema e, no final da execução, esta variável representa a energia elétrica consumida; a variável *power* representa a energia demandada inicial, e por fim, a variável *ava* representa a disponibilidade do modelo. A Figura 5.12 (b) mostra o conjunto de cores definido para o *token ini* do modelo CPN proposto. Esses valores representam a especificação de todos os atributos que o *token* carrega. O disparo da transição PS (ver Figura 5.11) consume o *token* do lugar 01, e a simulação continua a partir da subpágina da transição PS como mostrado na Figura 5.14.

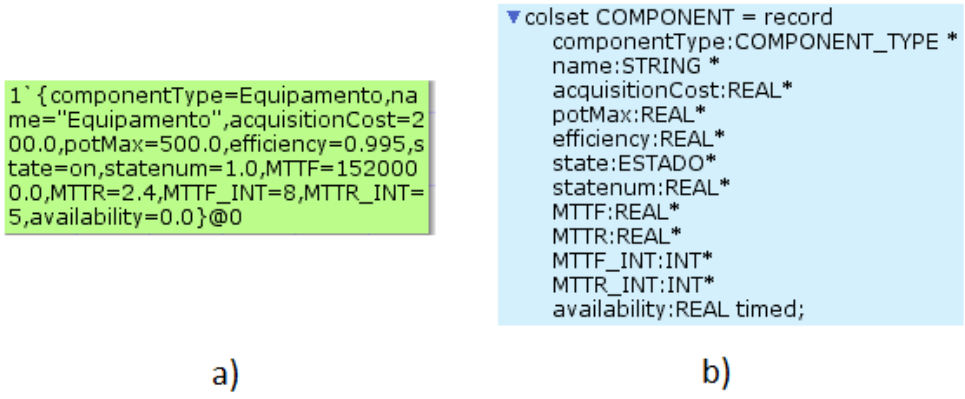


Figura 5.13: *Token* representando o equipamento/componente

Na Figura 5.11, além do *token* no lugar 01, existem outros *tokens* representados nos lugares ON e OFF que representam o estado inicial de cada componente do modelo. A Figura 5.13 (a) ilustra a representação da configuração inicial para o *token* utilizado para

representar um componente, no caso, o Power Strip. Vale ressaltar que cada equipamento tem os seus respectivos atributos específicos, como: nome e estado do equipamento, custo de aquisição, *MTTF* e *MTTR*, como também sua disponibilidade. A disponibilidade de toda a arquitetura é calculada após ser computado a disponibilidade de cada equipamento, sendo realizado ao final da execução. A Figura 5.13 (b) mostra a definição do conjunto de cor (colset) *COMPONENT* utilizado nos lugares que representam os componentes da arquitetura. Pode-se perceber que o conjunto de cores *COMPONENT* terá parâmetros para representar o tipo do componente (*componentType*), o nome (*name*), o preço (*acquisitionCost*), a potência máxima (*potMax*), a eficiência (*efficiency*), o estado (*state*), o MTTF e o MTTR, e a disponibilidade (*availability*).

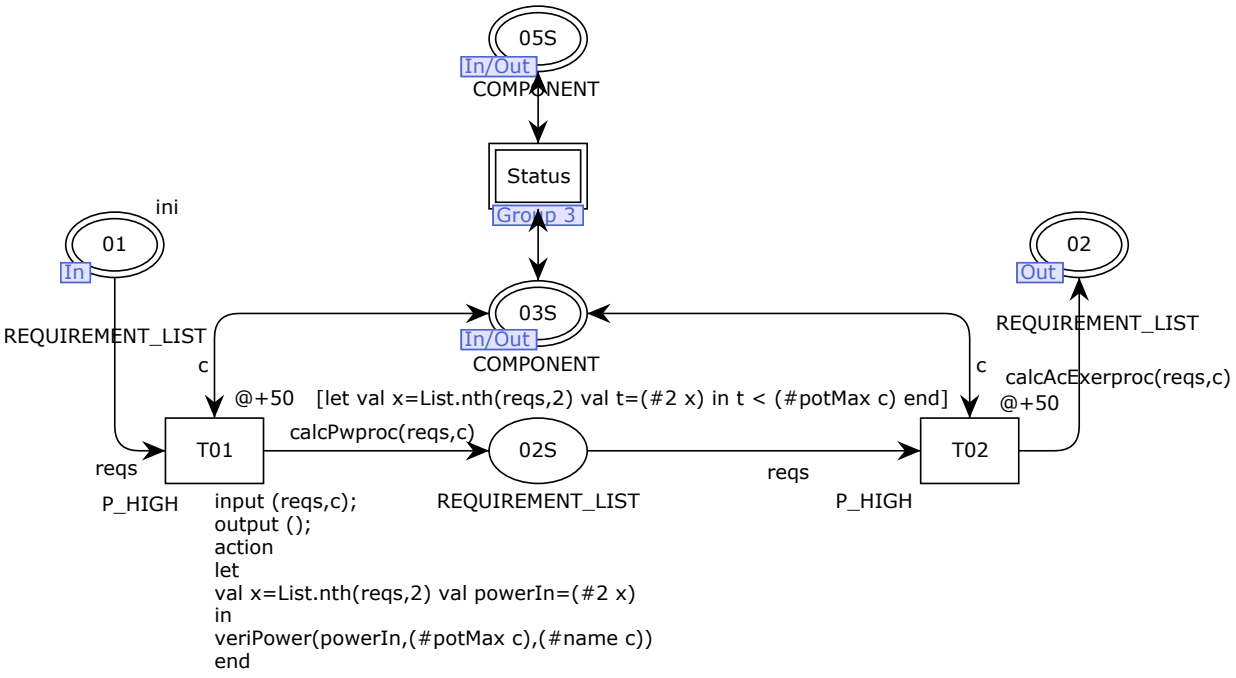


Figura 5.14: Modelo da subpágina da transição PS - componente Power Strip.

A Figura 5.14 mostra a subpágina da transição PS (ver Figura 5.11). Nesse modelo, a transição T01 estará habilitada quando existir um *token* no lugar 01 do tipo *REQUIREMENT_LIST* e outro no lugar 03S do tipo *COMPONENT*. Quando a transição T01 é disparada, um *token* do lugar 01 e outro do lugar 03S são consumidos. O leitor deve recordar que o fluxo energético só funciona se todos os dispositivos do caminho estiverem funcionando. Com isso, percebe-se a relação dinâmica do modelo de fluxo de energia com o modelo de disponibilidade dos equipamentos.

Ao acontecer o disparo da transição T01 (ver modelo da Figura 5.14), é chamada a função *calcPwproc()* que é responsável pelo cálculo da eficiência dos equipamentos (ver Equação 2.7). A função recebe como parâmetro um *token* do tipo $REQUIREMENT\_LIST$ (*reqs*) e um *token* do tipo $COMPONENT$ (c). A energia consumida vai variar de acordo com a eficiência dos equipamentos do modelo, tendo em vista que nenhum dispositivo elétrico tem uma eficiência energética de 100%. Sendo assim, energia é "perdida" ou dissipada ao longo do sistema.

A função *calcPwproc(reqs,c)* representa a chamada da função ilustrada pelo Algoritmo 1. Os parâmetros de entrada são os *tokens* com os valores específicos do fluxo de energia representado pelo *reqs* e do componente *c*. A primeira linha do algoritmo apresenta o nome da função, como também os parâmetros que recebe para realizar suas operações. Na segunda linha, é criada uma função local que fará a chamada de outra função (ver Algoritmo 2) para os cálculos da eficiência.

A terceira linha cria uma função local chamada *res* para realizar o mapeamento do tipo do nome da variável e o seu valor (ver Figura 5.12), e dessa forma, conseguir capturar o valor especifico que se deseja.

Algoritmo 1 Chamada do Cálculo da eficiência.

```
1: função CALCPWPROC(reqs,c)
2:     let val fun Sval(r) = calcPw(r, c)
3:     val res = (map Sval rs)
4:     in
5:     res
6:     End
7: fim função
```

A função *calcPw(r:REQUIREMENT,c:COMPONENT)* (ver Algoritmo 2) é chamada na linha 2 da função *calcPwproc()*. Essa função faz o cálculo da eficiência dos equipamentos fazendo a integração dos dados do fluxo de energia e dos atributos dos equipamentos, especificamente, da eficiência.

A primeira linha do algoritmo 2 apresenta o nome da função, como também os parâmetros que a função recebe. Na segunda e terceira linhas são criadas duas variáveis locais, a primeira é o nome do atributo e a segunda o seu valor (ver Figura 5.12 (a)).

Na linha 5, é verificado o nome do atributo que recebe o valor da divisão da energia demandada pela eficiência do equipamento. Por fim, a sétima e oitava linhas finalizam a função. A representação do Algoritmo 2 foi simplificada, onde somente a verificação da energia foi demonstrada. No entanto, outros testes similares as linhas 5 a 7 foram adicionados para verificar as demais métricas (ex., armazenar à eficiência individual de cada equipamento).

Algoritmo 2 Cálculo da eficiência.

```
1: função CALCPW(r:REQUIREMENT,c:COMPONENT)
2:     let val rtype = (#1 r)
3:      val rval = (#2 r)
4:     in
5:     caso rtype = power
6:         faça(rtype, rval/(#efficiency c))
7:     End
8: fim função
```

A Figura 5.15 mostra a subpágina da transição Status do submodelo da Figura 5.14. Nesta subpágina acontece o cálculo da disponibilidade, como também, a mudança do estado do equipamento (i.e., ON para OFF ou OFF para ON). Quando a transição T03 é disparada, a função *reevalStateOff()* faz com que o estado do equipamento fique em OFF, indicando que o dispositivo não está operacional. Já quando a transição T04 é disparada, a função *reevalStateOn()* realiza o cálculo da disponibilidade (ver Equação 2.2) e o equipamento muda para o status ON, que representa o estado operacional.

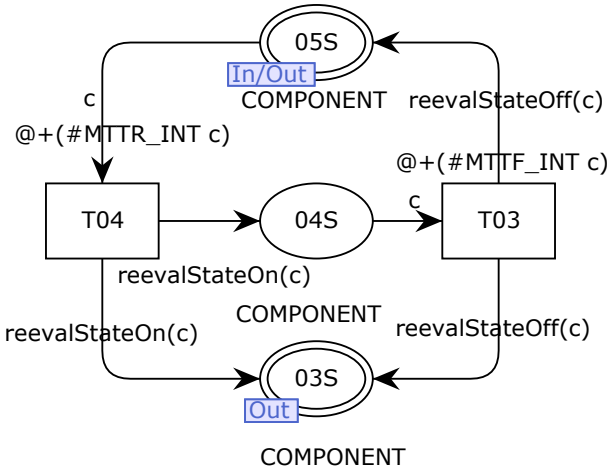


Figura 5.15: Transição Status

A verificação da capacidade máxima que cada componente pode suportar é necessária

para que o fluxo de energia não exceda a capacidade máxima de potência que cada equipamento pode prover. Esta verificação acontece, por exemplo, na transição T01 (ver Figura 5.14). Associada a essa transição existe uma função, *veriPower()*, escrita em CPNml que recebe como parâmetros: a energia demandada, a potência máxima que o equipamento suporta e o nome do equipamento. A função verifica se a energia que o equipamento está recebendo é maior do que a capacidade suportada. Se isso acontecer, a função registra em um arquivo (ver Figura 5.14 transição T01) qual equipamento não suportou o fluxo energético.

A função de guarda associada a transição T02, da subpágina PS (ver Figura 5.14), determina a condição na qual essa transição estará habilitada (i.e., fluxo de energia que passa pelo equipamento for menor do que a potência máxima que o equipamento suporta). Caso contrário, o fluxo será paralisado.

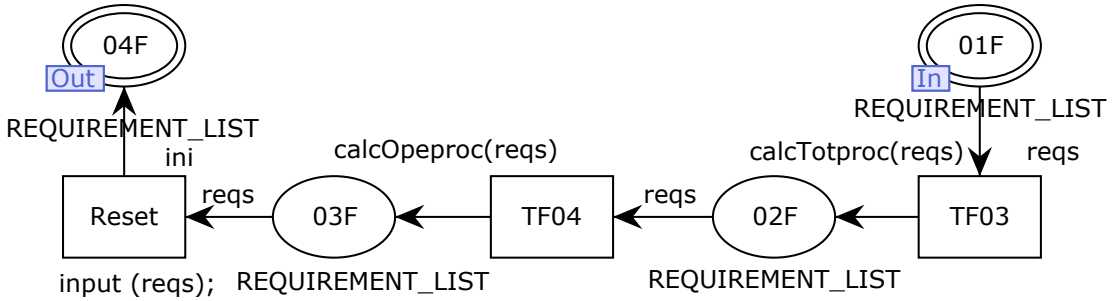
A função *reevalStateOn()*, localizada no arco da transição T04 ao lugar 04S da subpágina *Status* (ver Figura 5.15), é utilizada para computar a disponibilidade do equipamento e atualizar o estado operacional do mesmo dispositivo. O Algoritmo 3 representa essa função que recebe como entrada o *token* (do tipo *COMPONENT*) com os valores específicos desse dispositivo que está sendo avaliado. A primeira linha do algoritmo apresenta o nome da função, como também o parâmetro que a função recebe para realizar suas operações. Esse parâmetro c:COMPONENT é o *token* do equipamento. A segunda linha cria uma variável local, para receber o nome do componente. Na linha 4, acontece a condicional onde é verificado qual equipamento é escolhido para realizar as suas operações.

A linha 4 verifica qual o equipamento que vai ser analisado e, consequentemente, o *token* recebe os valores (i.e., custo de aquisição, potência máxima e eficiência energética) correspondentes ao dispositivo. A linha 6 finaliza a condicional. Posteriormente, os comandos das linhas 4 a 6 são repetidos para checar todos os equipamentos presentes no sistema modelado. A linha 7 mostra o retorno da função e, por fim, a linha 8 finaliza a função.

Algoritmo 3 Estado do equipamento On

```
1: função REEVALSTATEON(c)
2:     let val ctype=(#componentType c)
3:     in
4:     se ctype=NomeDoEquipamento então
5:         1`{componentType=NomeDoEquipamento, acquisitionCost=(#acquisitionCost c), potMax=(#potMax c), efficiency=(#efficiency c), state=not off, MTTF=(#MTTF c), MTTR=(#MTTR c), availability=(#MTTF c)/((#MTTR c)+(#MTTF c))}
6:         ...
7:     fim se
8:     retorna Token do tipo componente
9: fim função
```

A transição Final, localizada no modelo CPN da Figura 5.11, tem a função de calcular os parâmetros finais da simulação. A Figura 5.16 mostra o submodelo que representa o comportamento desta transição, onde o disparo da transição TF03 chama a função *calcTotproc()* que é ilustrada no Algoritmo 4.




```
input (reqs);
output ();
action
let val powerEnd = List.nth(reqs,2);
val powerEndd = (#2 powerEnd);
val meanPower=average(!meanEnergyList);
val powerError=standardError(!meanEnergyList);
val powerSD=standardDeviation(!meanEnergyList);
in
addIterData(powerEndd) ;
if checkEnd() then(
  isEnd:=1;
  writeFile1("\n\nMedia Global Da Energia: " ^ Real.toString meanPower^ "\n" ^
             "Desvio Padro da Energia: " ^ Real.toString powerSD ^ "\n" ^
             "Erro da Energia " ^ Real.toString powerError ^ "\n" ^
             "\n")
)
else
   newIter();
  isEnd:=0
end;
```

Figura 5.16: Transição Final

Os parâmetros de entrada são os *tokens* que carregam os valores finais do fluxo de energia e da disponibilidade da arquitetura. A variável *reqs* é do tipo REQUIREMENT_LIST. Na função está representada pelo parâmetro *rs*. A primeira linha do Algoritmo 4 apresenta o nome da função, como também o parâmetro que a função recebe para realizar suas operações. Na segunda linha, é criada uma função local que chama outra função (ver Algoritmo 5) para calcular a exergia operacional, o custo operacional, a eficiência energética da arquitetura, o *downtime* e a disponibilidade em número de noves (9s). A terceira linha cria uma variável local chamada *res* para realizar o mapeamento do tipo do nome da variável e o seu valor, e dessa forma, conseguir capturar o valor especifico que se deseja.

Algoritmo 4 Chamada do cálculo da exergia operacional, custo operacional, eficiência, downtime e 9s.

```
1: função calcTotproc(rs)
2:     fun Sval(r) = calcTot(r, rs)
3:     val res = (map Sval rs)
4:     in
5:     res
6:     End
7: fim função
```

A primeira linha do Algoritmo 5 apresenta o nome da função, como também os parâmetros que a função recebe para realizar suas operações. Na segunda e terceira linhas, são criadas duas variáveis locais, a primeira é o tipo do atributo e a segunda o seu valor (ver Figura 5.12 (a)).

Na linhas 4, 6, 8, 10 e 12 são realizadas operações na lista de atributos para buscar posições especificas, posições essas que representam os valores das variáveis que são carregadas pelo *token*.

Nas linhas 5, 7, 9, 11 e 13 acontecem a busca da posição 02 da tupla (ver Figura 5.12 (a)), são os atributos: custo de aquisição, exergia, energia inicial, energia demandada e disponibilidade respectivamente. A linha 15 representa a verificação de qual atributo que recebe o valor da métrica solicitada. A primeira métrica é o cálculo da exergia operacional (ver Equação 2.9) que envolve o somatório da exergia de todos os equipamentos, a disponibilidade da arquitetura e um período (pré-definido).

Algoritmo 5 Cálculo da exergia operacional, custo operacional, eficiência, downtime e 9s.

```
1: função calcTot(r:REQUIREMENT, lista:REQUIREMENT_LIST)
2:     let val rtype = (#1 r)
3:      val rval = (#2 r)
4:     val costAqu = List.nth(lista,0)
5:     val exergyReq = List.nth(lista,4)
6:     val exergyResu = (#2 exergyReq)
7:     val powerReqInI = List.nth(lista,1)
8:     val powerInIt = (#2 powerReqInI)
9:     val powerReq = List.nth(lista,2)
10:     val powerIn = (#2 powerReq)
11:     val ava1 = List.nth(lista,10)
12:     val avaEnd = (#2 ava1)
13:     in
14:     caso rtype = Exergia Operacional
15:         faça((rtype, rval/((exergyResu)*(avaEnd )*(!period)*(3.6/1000.0))
16:     caso rtype = Custo Operacional
17:         faça((rtype, !costEnergy*(!period)*(powerIn)))
18:     caso rtype = Eficiência Total
19:         faça((rtype,powerInIt/powerIn))
20:     caso rtype = Downtime
21:         faça((rtype, (1.0 - avaEnd)*(!period)))
22:     caso rtype = Números de 9
23:         faça((rtype, ((Math.log10 (1.0 - avaEnd)))*(-1.0)))
24:     End
25: fim função
```

A linha 17 verifica a variável que recebe o cálculo do custo operacional (ver Equação 2.6) que envolve o custo da energia (pré-definido), um período (pré-definido) e a energia elétrica consumida pela arquitetura.

A linha 19 verifica a variável que recebe o cálculo da eficiência da arquitetura (ver Equação 2.7) que envolve a energia elétrica demandada pelo sistema do *data center* e a energia de fato consumida decorrente da eficiência energética dos dispositivos.

A linha 21 verifica a variável que recebe o cálculo do *downtime* do sistema (ver Equação 2.8) que envolve o valor da disponibilidade da arquitetura e um período de tempo (pré-determinado). A linha 23 verifica a variável que recebe o cálculo dos números de 9s que envolve a disponibilidade da arquitetura.

A transição *Final*, localizada no modelo da Figura 5.11, tem a função de definir o critério de parada da simulação. O critério de parada adotado considera a precisão absoluta para

o consumo de energia. A precisão absoluta é calculada após a execução de 10 replicações do experimento. A precisão computada para o consumo de energia é comparada com os erros desejados. A simulação é concluída se esses valores calculados forem menores que a precisão desejada; caso contrário, a simulação prossegue calculando o número necessário de replicações.

Se as precisões desejadas foram alcançadas, o sistema exibe os resultados. Caso contrário, o fluxo de execução segue. Após a simulação, obtêm-se as métricas de interesse como o consumo de energia, a eficiência energética da arquitetura e a disponibilidade. Vale ressaltar que o algoritmo na linguagem CPNml que realiza essa operação está descrito na transição Reset (ver Figura 5.16).

O algoritmo na transição Reset (ver Figura 5.16) representa o processo adotado para o critério de parada, no qual o resultado do consumo de energia elétrica para cada execução do modelo CPN é incluído pela função *addIterData()*. Além disso, a função *checkEnd()* avalia se a simulação pode ou não continuar. Caso a simulação prossiga, outra função *newIter()* é executada para iniciar um novo ciclo com as variáveis reiniciadas. Caso contrário, a simulação é finalizada e a energia média, desvio padrão e o erro da energia são registrados em um arquivo pela função *writeFile1()*.

5.3 Modelo em paralelo

Na composição em paralelo (ver Figura 5.17), o sistema funcionará se um de seus componentes estiver funcionando. Portanto, este formato permite que mesmo na falha de um equipamento a arquitetura continue com o seu funcionamento.

A Figura 5.18 mostra a arquitetura de *data center* em paralelo que será usada para demostrar a aplicabilidade do formalismo proposto. Esta arquitetura tem os seguintes equipamentos: *Power Strip, Sub Panel, SDT, UPS1 e UPS2*. Essa arquitetura será modelada para ilustar o impacto da disponibilidade dos equipamentos no fluxo energético do sistema.

A Figura 5.19 (a) demostra um sistema simples de *data center* composto por 3 equipamentos, são eles: PowerStrip, UPS1 e UPS2. O UPS1 e UPS2 estão representando a

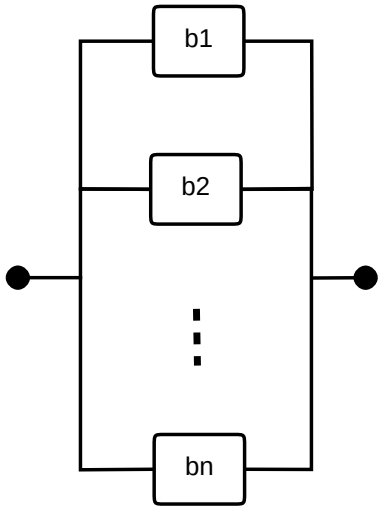


Figura 5.17: Sistema em paralelo.

redundância do modelo, ou seja, se uma dos caminhos vier a falhar, o fluxo energético passará pelo caminho que esteja com o equipamento operacional.

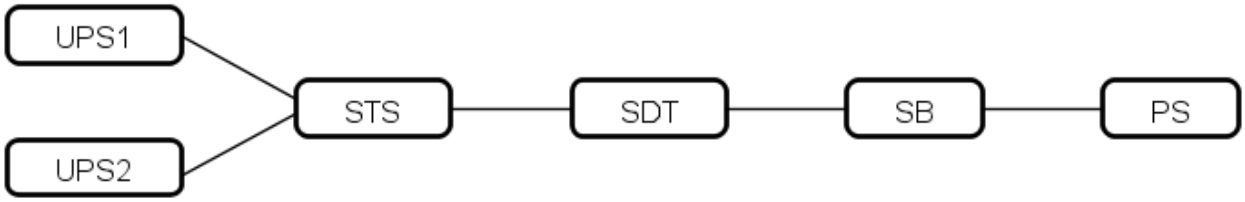


Figura 5.18: Arquitetura proposto em paralelo.

A Figura 5.19 (b) mostra o fluxo energético quando não existe falha elétrica em nenhum dos caminhos. Com a energia requerida de 100 kW, a energia de fato consumida pelo sistema será de 109,27 kW. Já a Figura 5.19 (c) mostra o fluxo energético com falha e a energia passando em apenas um caminho. Nesse cenário e assumindo a mesma demanda de energia de 100kW, o sistema consume 106,32kW.

Quando é identificado pelo modelo que um dos caminhos onde passará o fluxo energético se encontra inoperante, automaticamente todo o fluxo energético do sistema passará pelo caminho operacional. Usando como exemplo a Figura 5.18, existem dois caminhos para o fluxo energético. Um caminho partindo do UPS1 em direção ao PS e o outro a partir do UPS2 com o mesmo destino final.

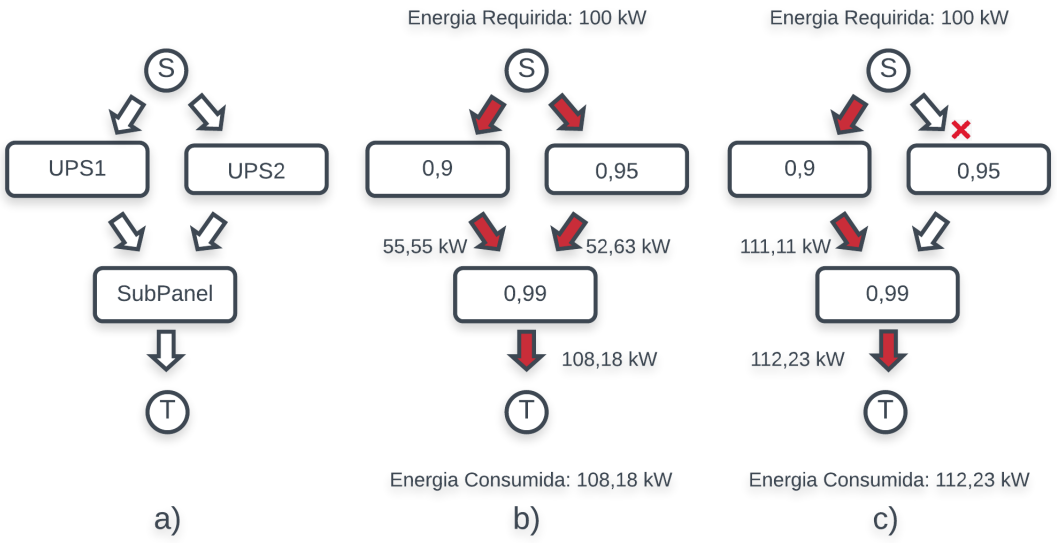


Figura 5.19: a) Sistema Simples; b) Sistema operante; c) Falha de um caminho.

Para exemplificar a escolha do caminho, foi implementado na ferramenta CPNTools a arquitetura mostrada na Figura 5.18 e, ilustrado aqui, somente apenas o momento da escolha dos caminhos. A Figura 5.20 mostra o momento da escolha de um dos caminhos a ser tomado pelo fluxo energético, decorrente da inoperância de um dos fluxos. A transição TD01 quando disparada aciona as funções *calcWeiproc1()* e *calcWeiproc2()*, que recebem como parâmetros: *reqs*, *c* e *c1* do tipo REQUEREMENT_LIST, COMPONENT e COMPONENT respectivamente. Todos possuem um conjunto de informações referentes ao fluxo energético e disponibilidade dos componentes da parte superior e da parte inferior. Informações que podem ser carregadas a partir dos componentes, por exemplo, podem ser responsáveis pela verificação do funcionamento operacional do equipamento.

A partir disso, é verificado quais os equipamentos estão inoperantes para direcionar o fluxo energético pelo caminho operacional (se existir). Essas funções também são responsáveis por realizar a distribuição do fluxo energético de acordo com o peso presente nos arcos.

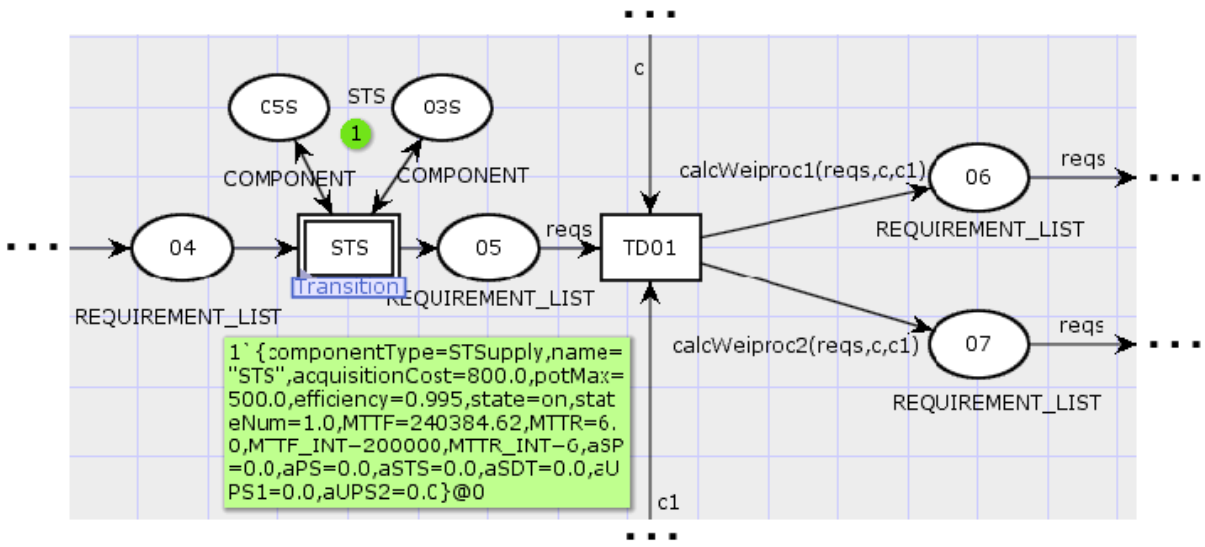


Figura 5.20: Verificação do fluxo energético

5.4 Resumo

Este capítulo apresentou o modelo proposto em CPN. No primeiro momento foi mostrado um modelo simples com apenas um equipamento, e também, demonstrou a importância do processo de hierarquia das transições nos modelos CPNs. No segundo momento foi apresentado o modelo da arquitetura A1 em série, bem como, demostradas todas as funções dos cálculos do modelo. No terceiro momento foi exemplificado o modelo para arquitetura em paralelo.

Capítulo 6

Estudo de Caso

Este capítulo apresenta dois estudos de caso que têm como objetivo ilustrar a aplicabilidade da metodologia proposta em um cenário real da infraestrutura elétrica de um *data center* típico. O primeiro estudo de caso foca a validação dos modelos propostos. O segundo estudo analisa o consumo energético de uma arquitetura de *data center* levando em consideração a diminuição do MTTF de apenas um equipamento da arquitetura. O foco do segundo estudo é ilustrar o impacto da disponibilidade no fluxo energético.

6.1 Estudo de Caso I

O estudo de caso foca a validação dos modelos propostos através da análise de cenários reais. A Figura 6.1 mostra as arquiteturas que serão avaliadas neste estudo de caso. A arquitetura A1 representa o caminho do fluxo em série e sem nenhuma redundância. A partir da arquitetura A2 equipamentos foram adicionados para aumentar a disponibilidade, de modo que cada arquitetura sucessiva tenha um componente adicional duplicado. Os resultados desta avaliação serão comparados com os resultados obtidos por [58]. Vale salientar que no estudo de caso foi considerado a arquitetura proposta A1 em pleno funcionamento do ponto de vista energético, não tendo a disponibilidade afetado a arquitetura.

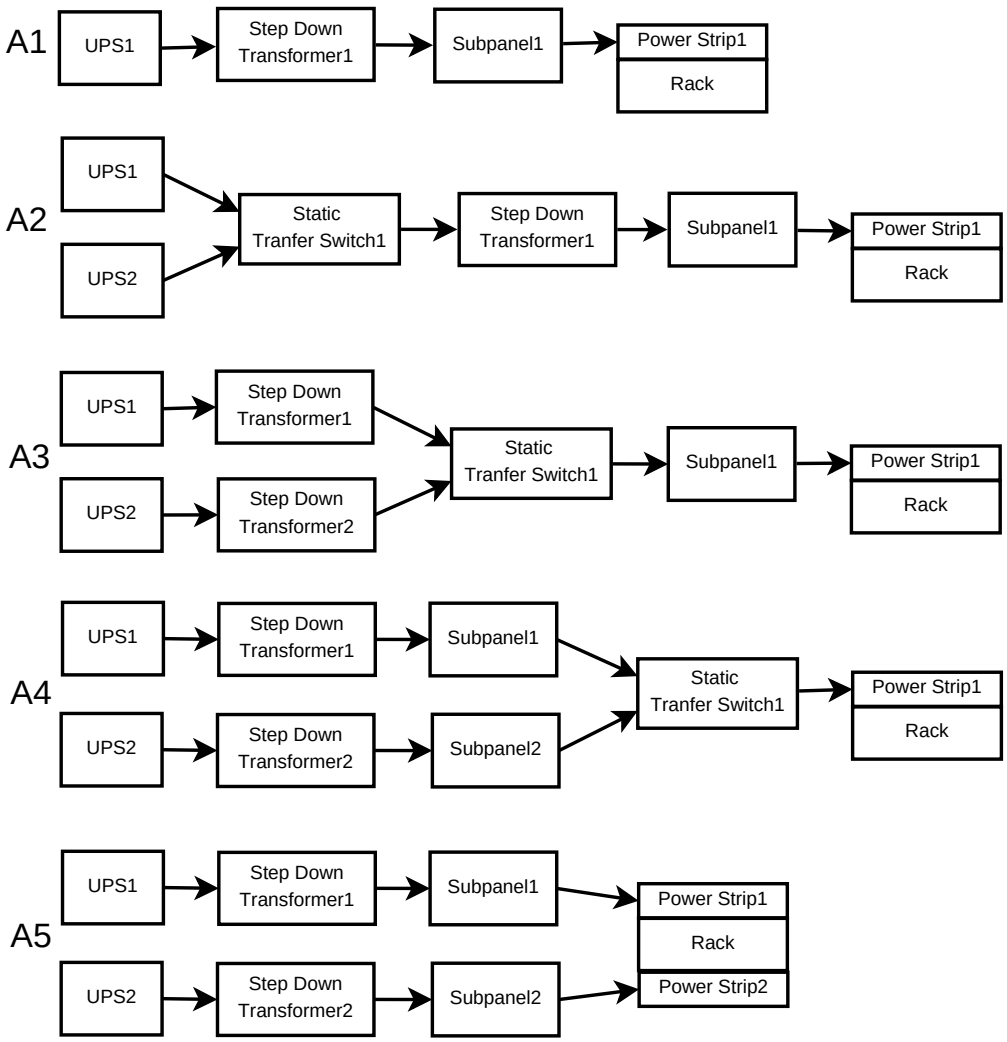


Figura 6.1: Arquitetura de *data center*

Os valores de MTTFs, MTTRs, custo de aquisição e a eficiência energética para cada equipamento foram retirados de [59, 60, 61] e são mostrados na Tabela 6.1.

6.1.1 Resultados

Para execução dos cálculos foram atribuídos valores aos parâmetros de entrada, como 100kW para o valor da energia demandada pelos equipamentos de TI. Para o custo da energia elétrica foi considerado o valor de *BRL* 0,11 e o tempo de 5 anos, ou seja, 43800 horas.

Tabela 6.1: Valores de MTTF, MTTR, Custo de aquisição e eficiência

Componentes	MTTF (h)	MTTR (h)	Aquisição (BRL)	Eficiência (%)
UPS	250.000,00	8,00	3600,00	95,3
SDT	1.412.908,33	156,01	550,00	98,5
Sub Panel	1.520.000,00	2,40	200,00	99,5
Power Strip	11.511.175,63	3,80	200,00	99,5
STS	240.384,62	6,00	800,00	99,5
ATS	500.000,00	0,33	800,00	99,5

A Tabela 6.2 apresenta um resumo dos resultados da avaliação das 5 arquiteturas, onde 9s representa a disponibilidade em número de noves calculada usando $-log(1-disp(\%)/100)$. Como esperado, a disponibilidade aumenta quando componentes mais redundantes são inseridos. Na mesma tabela também é apresentado um resumo dos resultados da avaliação para: Exergia operacional e energia consumida. A exergia operacional é utilizada para identificar o sistema elétrico com maior eficiência energética.

Essa Tabela 6.2 mostra ainda os resultados obtidos com o modelo proposto para as métricas de: custo de aquisição, custo operacional e custo total da arquitetura de *data center*. O custo de aquisição corresponde aos recursos necessários para implementar a infraestrutura do *data center*. O custo operacional é o custo para manter o sistema no modo operacional. Os resultados obtidos para cada uma dessas métricas são exatamente iguais aos resultados obtidos por outros autores em [58], e que faziam uso de outras técnicas de modelagem (SPN, RBD e EFM). Sendo assim, o modelo CPN proposto foi validado. É importante destacar que, diferentemente do [58], este trabalho é capaz de realizar os cálculos dinamicamente, ou seja, é computado o consumo final da energia elétrica levando em consideração a disponibilidade dos equipamentos. Dessa forma, a utilização desses modelos para quantificar o impacto da disponibilidade em tempo de execução em arquiteturas de *data centers* faz com que os projetistas possam prever o consumo energético antes da implementação, como também identificar o consumo final de energia elétrica levando em consideração a disponibilidade dos equipamentos que compõem a estrutura. Vale destacar que, com o acréscimo de equipamentos redundantes, o sistema apresentou um aumento na disponibilidade de forma significativa

(ver Tabela 6.2) com praticamente o mesmo custo operacional e o mesmo impacto ambiental.

Tabela 6.2: Resumo dos resultados.

Arquitetura	Disp. (%) (9s)	Efi. (%)	Exergia Ope. (GJ)	Ener. Consumida (kW)	C. de Aquisição (BRL)	C. Operacional (BRL)	C. Total (BRL)
A1 (U-T-P-S)	99,98556 (3,84)	93,3	1198,68	107,60	4550,00	518.356,83	522.906,83
A2 (2U-STS-T-P-S)	99,98627 (3,86)	92,8	1283,67	108,14	8950,00	520.965,30	529.915,30
A3 (2U-2T-STS-P-S)	99,99731 (4,57)	92,8	1284,00	108,14	9500,00	521.022,82	530.522,82
A4 (2U-2T-2P-STS-S)	99,99989 (5,99)	92,8	1284,11	108,14	9700,00	521.036,30	530.736,30
A5 (2U-2T-2P-2S)	99,99999 (7,68)	93,3	1198,85	107,60	9100,00	518.431,63	527.531,63

U-UPS; T - Transformer; P - SubPanel; S - Rack power strip; STS - Static transfer switch

6.2 Estudo de Caso II

O objetivo desse estudo de caso é demostrar o impacto da disponibilidade no fluxo energético de arquiteturas de *data centers*. Dessa forma, será demostrado o quanto a disponibilidade dos equipamentos das arquiteturas analisadas de fato impactam no fluxo energético do sistema.

6.2.1 Arquiteturas analisadas

As arquiteturas que foram analisadas para o segundo estudo de caso são: A1 à A5 (ver Figura 6.1). A arquitetura A1 está construída na disposição em série, ou seja, se qualquer um dos dispositivos deste modelo ficar indisponível todo o sistema do ponto de vista elétrico estará indisponível.

Dessa forma, a análise do modelo CPN proposto que representa essa arquitetura A1 mostra dois resultados possíveis com relação a energia elétrica. São eles: (i) todos os equipamentos se encontram em funcionamento; ou (ii) quando qualquer equipamento estiver indisponível, ou seja, o sistema como um todo falho. A partir da arquitetura A2, equipamentos foram adicionados para aumentar a disponibilidade do sistema. Vale destacar que ao aumentar a possibilidade de caminhos redundantes, também aumenta os resultados possíveis para o nosso modelo proposto e, principalmente, o impacto da disponibilidade no fluxo energético.

6.2.2 Resultados

A Figura 6.2 mostra a relação do impacto da disponibilidade no fluxo energético, resultando em um menor consumo final de energia elétrica pela arquitetura. A primeira barra mais escura do gráfico, mostra o consumo energético da arquitetura A1 sem a integração com a disponibilidade. A partir da segunda barra é feita a comparação do consumo energético quando a disponibilidade afeta o fluxo de energia elétrica. O MTTF de apenas um equipamento (UPS) é diminuído para a verificação do consumo de energia e, consequentemente, o consumo elétrico é menor, devido ao equipamento ficar inoperante por mais tempo, e ficando

ocioso, do ponto de vista energético.

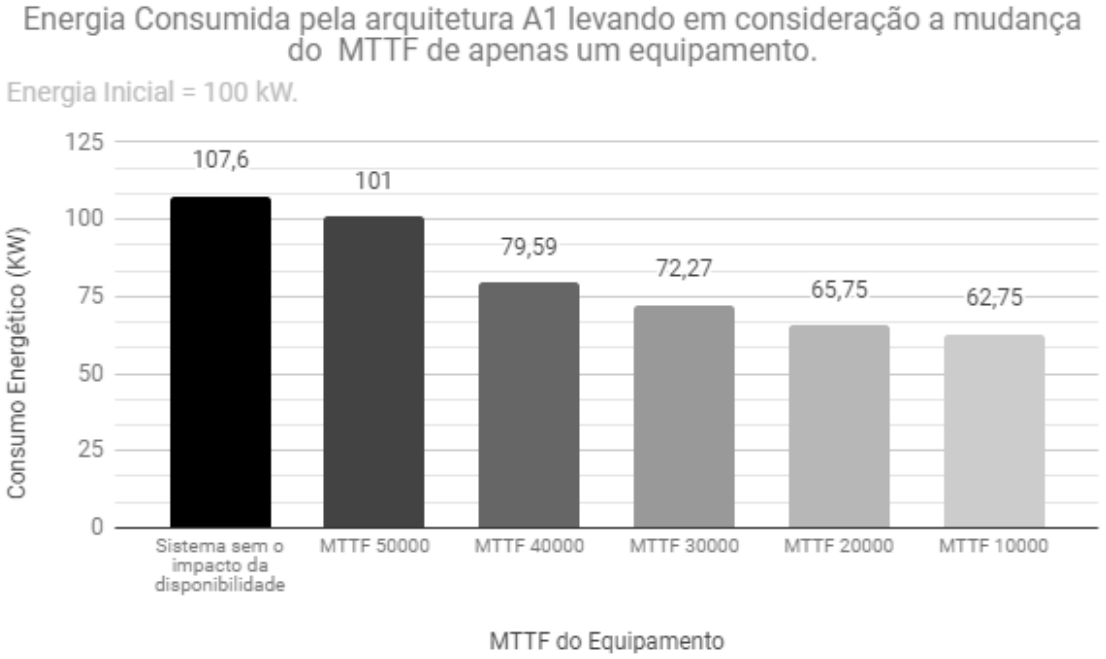
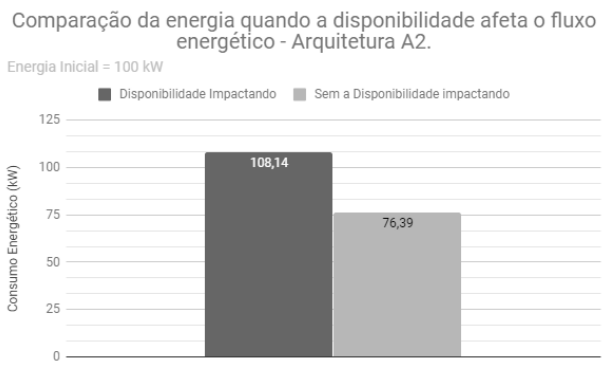


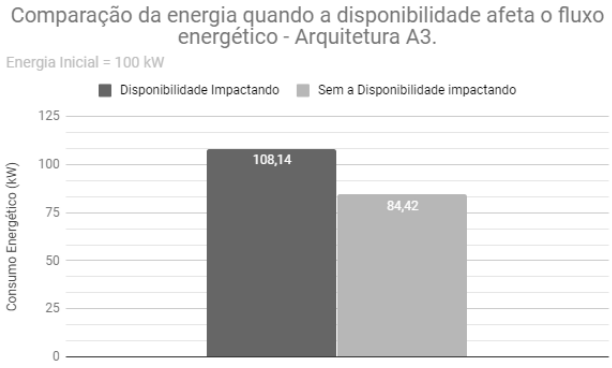
Figura 6.2: Consumo Energético - Arquitetura A1.

A Figura 6.3 (a), (b), (c) e (d) mostram os resultados do consumo de energia elétrica das arquiteturas A2, A3, A4 e A5 levando em consideração dois cenários: (i) sem a disponibilidade impactar o fluxo energético, e (ii) com a disponibilidade impactando o fluxo energético do *data center*. A primeira barra com a cor mais escura de todos os gráficos mostram a energia consumida quando a disponibilidade não afeta o sistema, já a segunda barra com a cor mais clara demostram o impacto da disponibilidade no sistema.

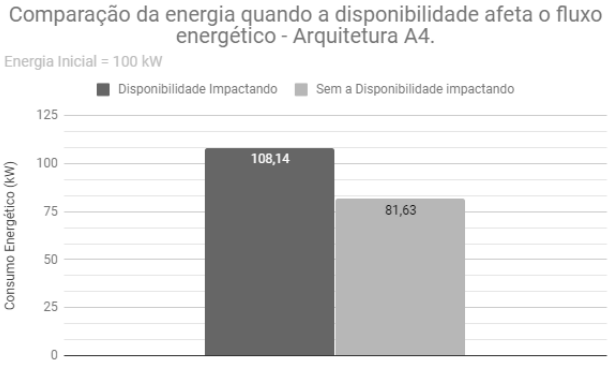
A Figura 6.3 (a), mostra um consumo de 108,14 kW de energia quando a disponibilidade não afeta a arquitetura, no momento em que a disponibilidade começa a afetar a arquitetura o consumo de energia cai para 76.39 kW. Esta queda no consumo também ocorre nas arquiteturas A3, A4 e A5, mostrados nas Figuras 6.3 (b), (c) e (d) respectivamente. A Tabela 6.3 mostra todos os resultados para as arquiteturas mencionadas. Pode-se concluir que, durante o processo de simulação, se algum dos componentes elétricos apresentar inoperância o consumo de energia deste sistema será menor, se comparado com o sistema sem a presença de falhas nos componentes elétricos.



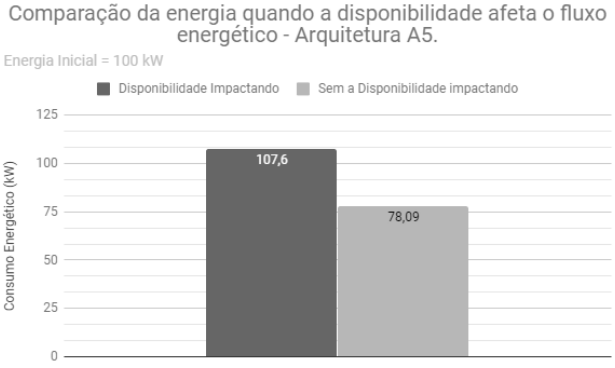
(a) Consumo Energético - Arquitetura A2



(b) Consumo Energético - Arquitetura A3



(c) Consumo Energético - Arquitetura A4



(d) Consumo Energético - Arquitetura A5

Figura 6.3: Consumo energético da arquitetura A2 à A5.

Tabela 6.3: Resumo do consumo energético das arquiteturas A1, A2, A3, A4 e A5.

Arquitetura	Disponibilidade não afetando (kW)	Disponibilidade afetando (kW)
A1	107,60	101,00
A2	108,14	76.39
A3	108,14	84.42
A4	108,14	81.63
A5	107,60	78.09

6.3 Resumo

Este capítulo apresentou dois experimentos para avaliar os modelos propostos e, também, para demonstrar a importância da disponibilidade no processo de análise do fluxo energético. O primeiro exemplificou o modelo para focar na validação. O segundo experimento foi realizado para ilustrar uma comparação do consumo energético, em tempo de execução, quando a disponibilidade afeta a arquitetura do ponto de vista energético.

Capítulo 7

Conclusão

Os serviços em nuvem estão cada vez mais difundidos e utilizados pelos usuários da grande rede, tendo demando um aumento na capacidade de processamento e armazenamento de dados dos *data centers*. Para que esses serviços, atingissem altos níveis de disponibilidade exigidos pela computação em nuvem, foi necessário que os *data centers* evoluíssem, porém, esses altos índices de disponibilidade impactam diretamente na sustentabilidade. Portanto, as preocupações sobre confiabilidade, sustentabilidade e custo dos sistemas de *data center* estão em foco tanto pela comunidade acadêmica quanto pela sociedade

Sendo assim, este trabalho propõe uma estratégia integrada e dinâmica que demostra o impacto da disponibilidade dos equipamentos que compõem a arquitetura de *data centers* no consumo energético. Foi utilizado a técnica de modelagem CPN para quantificar o custo, impacto ambiental e a disponibilidade da infraestrutura de energia elétrica de *data center*. As arquiteturas foram implementadas no ambiente CPNTools, utilizando a linguagem de programação CPN ML para fornecer os dados solicitados.

Foram analisados dois estudos de caso:

- No primeiro, foram implementadas cinco arquiteturas elétricas básicas de *data center*. Todos os resultados obtidos das cinco arquiteturas foram validados através de trabalhos anteriores que utilizaram outras técnicas de modelagem, como: SPN, RBD e EFM. Os resultados foram validados separadamente, pois, o presente trabalho realiza os

cálculos da disponibilidade de forma dinâmica com os cálculos de fluxo de energia. Nos trabalhos anteriores para se obter os resultados de disponibilidade das arquiteturas foi utilizado o formalismo RBD e/ou SPN e para as métricas de fluxo de energia o EFM.

- Já o segundo estudo, teve o objetivo de demostrar o impacto da disponibilidade no fluxo energético, dessa forma, foi demostrado através dos modelos propostos o quanto a disponibilidade dos equipamentos afetam o resultado do fluxo energético da arquitetura.

Pode-se concluir que a utilização de estratégias com a criação de modelos para quantificar o impacto da disponibilidade em tempo de execução em arquiteturas de *data centers* faz com que os projetistas possam prever o consumo energético antes da implementação, como também identificar o consumo final de energia elétrica levando em consideração a disponibilidade dos equipamentos que compõem a estrutura.

7.1 Contribuições

Este trabalho propôs uma estratégia integrada e dinâmica para obtenção de métricas de infraestruturas de *data center*, considerando questões de confiabilidade, sustentabilidade e custo. Além de tudo, a metodologia proposta permite a avaliação de arquiteturas simples, como também, arquiteturas com uma disposição complexa com componentes redundantes de *data center*.

Vale ressaltar que esse é o primeiro trabalho a utilizar rede de Petri colorida na quantificação do impacto da disponibilidade no fluxo de energia, onde a disponibilidade impacta diretamente nos resultados do consumo de energia, por exemplo. A disponibilidade é verificada em tempo real, ou seja, em tempo de execução. Além disso, o modelo proposto para a quantificação do impacto ambiental, é capaz de computar o custo e fazer a verificação da capacidade máxima de potência que cada equipamento pode prover.

Para um melhor entendimento, todas as contribuições desta pesquisa estão especificadas abaixo:

- **Metodologia:** Através da metodologia que foi adotada permite construir arquiteturas com a integração das métricas de disponibilidade e das métricas do modelo de fluxo de energia. Dessa forma, é possível computar a disponibilidade, o custo e a exergia dos equipamentos utilizando apenas um formalismo de forma integrada e dinâmica;
- **Modelagem:** Os modelos CPN foram propostos para computar métricas de disponibilidade, sustentabilidade e custo de diversas arquiteturas elétricas de *data centers*;
- **Integração dinâmica:** Foi proposto um modelo CPN capaz de computar as métricas de disponibilidade e fluxo de energia. Para tal, foi utilizada apenas uma técnica de modelagem para essa integração e consequentemente a obtenção das métricas de interesse. Dessa forma, quando um equipamento da arquitetura apresenta algum tipo de falha, os resultados finais desta arquitetura sofrerão impactos.

7.2 Trabalhos Futuros

A aplicabilidade destra trabalho poderá ser implementada para arquiteturas de *data centers* maiores e mais complexas, já que utiliza um paradigma que pode ser facilmente modelado às necessidades dos projetistas.

Como futuros direcionamentos, espera-se estender os modelos propostos em CPN para adicionar novas funcionalidades como ,por exemplo, quantificar o consumo energético de serviços hospedados nos *data centers*. Isso pode ser analisado com os recursos providos pelas CPNs e que não estavam disponíveis nos modelos EFMs e nem nos modelos normalmente utilizados para a quantificação da disponibilidade como os diagrama de blocos de confiabilidade e redes de Petri estocásticas.

Outras possibilidades de trabalhos futuros:

- Avaliar/Explorar outras arquiteturas de *data center* como: TIER 1, 2, 3 e 4;
- Propor uma ferramenta gráfica para facilitar a modelagem e avaliação das arquiteturas propostas;

- Propor a integração de uma ferramenta que compute os resultados de CPN com outros formalismos;
- Propor métodos de otimização para executar com o modelo proposto;
- Propor uma extensão da ferramenta para gerar de forma automática modelos em CPN.

7.3 Limitações

Este trabalho foi desenvolvido com a técnica de formalismo CPN, cuja complexidade do modelo, necessita que os leitores tenham um conhecimento prévio do formalismo.

Referências Bibliográficas

[1] B. Weihl, E. Teetzel, J. Clidaras, C. Malone, J. Kava, and M. Ryan, "Sustainable data centers," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 17, no. 4, pp. 8–12, 2011.

[2] S. Kemp, "Digital in 2018: World's internet users pass the 4 billion mark," *We are social*, 2018.

[3] C. Low, Y. Chen, and M. Wu, "Understanding the determinants of cloud computing adoption," *Industrial management & data systems*, vol. 111, no. 7, pp. 1006–1023, 2011.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] N. Leavitt, "Is cloud computing really ready for prime time?," *Computer*, no. 1, pp. 15–20, 2009.

[6] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, "Availability and load balancing in cloud computing," in *International Conference on Computer and Software Modeling, Singapore*, vol. 14, 2011.

[7] "Microsoft - creating a greener data center." `http://www.microsoft.com/presspass/features/2009/apr09/04-02Greendatacenters.mspx`, 2009.

[8] A. S. Andrae and T. Edler, "On global electricity usage of communication technology: trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.

[9] M. Poess and R. O. Nambiar, "Energy cost, the key challenge of today's data centers: A power consumption analysis of tpc-c results," *Proc. VLDB Endow.*, vol. 1, pp. 1229–1240, Aug. 2008.

[10] R. Buyya, C. Vecchiola, and S. T. Selvi, *Mastering Cloud Computing: Foundations and Applications Programming.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2013.

[11] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, (Berkeley, CA, USA), pp. 3–3, USENIX Association, 2008.

[12] G. Callou, P. Maciel, D. Tutsch, and J. Araújo, "Models for dependability and sustainability analysis of data center cooling architectures," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*, pp. 1–6, IEEE, 2012.

[13] G. Callou, P. Maciel, D. Tutsch, J. Ferreira, J. Araújo, and R. Souza, "Estimating sustainability impact of high dependable data centers: a comparative study between brazilian and us energy mixes," *Computing*, vol. 95, no. 12, pp. 1137–1170, 2013.

[14] A. M. Law, W. D. Kelton, and W. D. Kelton, *Simulation modeling and analysis*, vol. 2. McGraw-Hill New York, 1991.

[15] A. Headquarters, "Cisco data center infrastructure 2.5 design guide," *Cisco Validated Design I. Cisco Systems, Inc*, 2007.

[16] "The equipment energy efficiency (e3) program, "energy efficiency policy options for australian and new zealand data centres"," 2014.

[17] C. L. Belady, "In the data center, power and cooling costs more than the it equipment it supports," *Electronics cooling*, vol. 13, no. 1, p. 24, 2007.

[18] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH computer architecture news*, vol. 35, pp. 13–23, ACM, 2007.

[19] M. Arregoces and M. Portolani, *Data center fundamentals*. Cisco Press, 2003.

[20] V. K. Katukoori, "Standardizing availability definition," *University of New Orleans, New orleans, La., USA*, 1995.

[21] P. R. Maciel, K. S. Trivedi, R. Matias, and D. S. Kim, "Dependability modeling," *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, vol. 1, pp. 53–97, 2011.

[22] P. R. Maciel, K. S. Trivedi, R. Matias, and D. S. Kim, "Dependability modeling," in *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, pp. 53–97, IGI Global, 2012.

[23] I. Dincer, "Thermodynamics, exergy and environmental impact," *Energy sources*, vol. 22, no. 8, pp. 723–732, 2000.

[24] D. Schmidt, "Low exergy systems for high-performance buildings and communities," *Energy and Buildings*, vol. 41, no. 3, pp. 331–336, 2009.

[25] C. A. Petri, "Communicating with automata," *Germany: PhD thesis, Technical University Darmstadt*, 1962.

[26] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[27] P. Merlin and D. Farber, "Recoverability of communication protocols–implications of a theoretical study," *IEEE transactions on Communications*, vol. 24, no. 9, pp. 1036–1043, 1976.

[28] M. A. Marsan, "Stochastic petri nets: an elementary introduction," in *European Workshop on Applications and Theory in Petri Nets*, pp. 1–29, Springer, 1988.

[29] K. Jensen, "Coloured petri nets: A high level language for system design and analysis," in *International Conference on Application and Theory of Petri Nets*, pp. 342–416, Springer, 1989.

[30] V. Janoušek, *Modelling Objects by Petri Nets*. PhD thesis, PhD. thesis, Brno University of Technology, Brno, Czech Republic, 1998.

[31] K. Jensen, *Coloured Petri nets: basic concepts, analysis methods and practical use*, vol. 1. Springer Science & Business Media, 2013.

[32] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured petri nets and cpn tools for modelling and validation of concurrent systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3-4, pp. 213–254, 2007.

[33] R. Milner, *The definition of standard ML: revised*. MIT press, 1997.

[34] P. Huber, K. Jensen, and R. M. Shapiro, "Hierarchies in coloured petri nets," in *International Conference on Application and Theory of Petri Nets*, pp. 313–341, Springer, 1989.

[35] K. Jensen, "A brief introduction to coloured petri nets," in *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 203–208, Springer, 1997.

[36] L. M. Kristensen, S. Christensen, and K. Jensen, "The practitioner's guide to coloured petri nets," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 2, no. 2, pp. 98–132, 1998.

[37] K. Jensen and L. M. Kristensen, *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009.

[38] R. Harper, "Programming in standard ml," 2001.

[39] P. R. Maciel, R. D. Lins, and P. R. Cunha, *Introdução às redes de Petri e aplicações*. UNICAMP-Instituto de Computacao, 1996.

[40] R. Pucella, "Notes on programming standard ml of new jersey," *Cornell University, January*, 2001.

[41] E. Németh, R. Lakner, K. Hangos, and I. Cameron, "Hierarchical cpn model-based diagnosis using hazop knowledge," *Research Report*, 2003.

[42] G. Callou, P. Maciel, E. Tavares, E. Andrade, B. Nogueira, C. Araujo, and P. Cunha, "Energy consumption and execution time estimation of embedded system applications," *Microprocessors and Microsystems*, vol. 35, no. 4, pp. 426–440, 2011.

[43] É. Rocha, P. T. Endo, G. Leoni, J. Braga, and T. Lynn, "Analyzing the impact of power infrastructure failures on cloud application availability," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pp. 1746–1751, IEEE, 2017.

[44] Q. Wang, X. Wang, and S. Yang, "Energy modeling and simulation of flexible manufacturing systems based on colored timed petri nets," *Journal of Industrial Ecology*, vol. 18, no. 4, pp. 558–566, 2014.

[45] E. Andrade, B. Nogueira, R. Matos, G. Callou, and P. Maciel, "Availability modeling and analysis of a disaster-recovery-as-a-service solution," *Computing*, pp. 1–26, 2017.

[46] M. Talebberrouane, F. Khan, and Z. Lounis, "Availability analysis of safety critical systems using advanced fault tree and stochastic petri net formalisms," *Journal of Loss Prevention in the Process Industries*, vol. 44, pp. 193–203, 2016.

[47] C. Melo, R. Matos, J. Dantas, and P. Maciel, "Capacity-oriented availability model for resources estimation on private cloud infrastructure," in *Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium on*, pp. 255–260, IEEE, 2017.

[48] E. Sousa, F. Lins, E. Tavares, P. Cunha, and P. Maciel, "A modeling approach for cloud infrastructure planning considering dependability and cost requirements," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 4, pp. 549–558, 2015.

[49] T. A. Nguyen, D. Min, E. Choi, and T. D. Tran, "Reliability and availability evaluation for cloud data center networks using hierarchical models," *IEEE Access*, vol. 7, pp. 9273–9313, 2019.

[50] M. Wiboonrat, "An empirical study on data center system failure diagnosis," in *Internet Monitoring and Protection, 2008. ICIMP'08. The Third International Conference on*, pp. 103–108, IEEE, 2008.

[51] Y. Liu, X. Li, Y. Lin, R. Kang, and L. Xiao, "A colored generalized stochastic petri net simulation model for service reliability evaluation of active-active cloud data center

based on it infrastructure," in *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, pp. 51–56, IEEE, 2017.

[52] B. Pinna, G. Babykina, N. Brinzei, and J.-F. Pétin, "Using coloured petri nets for integrated reliability and safety evaluations," *IFAC Proceedings Volumes*, vol. 46, no. 22, pp. 19–24, 2013.

[53] H. Song, "Modeling of railway system maintenance and availability by means of colored petri nets modelowanie utrzymania ruchu i gotowości systemu kolejowego za pomoca kolorowych sieci petriego," *EKSPLOATACJA I NIEZAWODNOSC*, vol. 20, no. 2, p. 236, 2018.

[54] Z. Wang, M. Atli, and H. Kondo Adjallah, "Coloured stochastic petri nets modelling for the reliability and maintenance analysis of multi-state multi-unit systems," *Journal of Manufacturing Technology Management*, vol. 25, no. 4, pp. 476–490, 2014.

[55] P. Vrignat, M. Avila, F. Duculty, S. Aupetit, M. Slimane, and F. Kratz, "Maintenance policy: degradation laws versus hidden markov model availability indicator," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 226, no. 2, pp. 137–155, 2012.

[56] D. F. Fitch and H. Xu, "A petri net model for secure and fault-tolerant cloud-based information storage.," in *SEKE*, pp. 333–339, Citeseer, 2012.

[57] C. A. Chung, *Simulation modeling handbook: a practical approach*. CRC press, 2003.

[58] G. R. d. A. Callou, "Assessment to support the planning of sustainable data centers with high availability," 2013.

[59] *IEEE Gold Book 473, Design of Reliable Industrial and Commercial Power Systems.* 2017.

[60] V. Avelar, "Comparing availability of various rack power redundancy configurations," *APC White Paper*, vol. 48, pp. 1–22, 2003.

[61] Hewlett-Packard, "Hp power advisor tool.." `http://h18004.www1.hp.com/products/solutions/power`, 2013.