Desenvolvimento de Uma Ferramenta para Monitoramento de Softwares Web

Tonin De Rosso Bolzan, Sylvio André Garcia Vieira

Centro Universitário Franciscano (UNIFRA) – Santa Maria, RS – Brasil

tonin@bolzan.io, sylvio@unifra.br

Abstract. *The availability and integrity of softwares developed for the internet are essential to satisfy the requirements of its users and owners. However, the supervision of these factors is not a simple task when performed by humans, especially when multiple softwares should be supervised at the same time. Therefore, this work contemplates developing of a monitoring tool that makes possible the implementation of supervision periodically and continuously. The results of the tool helped to identify promptly when the monitored sites have changed in their activities.*

Resumo. *A disponibilidade e integridade dos softwares desenvolvidos para internet são de fundamental importância para satisfazer os requisitos de seu usuários e proprietários. No entanto, a supervisão desses fatores não é uma tarefa simples quando exercida por seres humanos, principalmente quando vários softwares devem ser supervisionados ao mesmo tempo. Por isso, este trabalho contempla o desenvolvimento de uma ferramenta de monitoramento que torna viável a execução dessa supervisão de forma periódica e contínua. Os resultados da ferramenta permitiram identificar, prontamente, quando os sítios monitorados sofreram alterações em suas atividades.*

1. Introdução

Segundo uma pesquisa realizada pela Netcraft[1] (2014), existem mais de 861 milhões de servidores web em atividade atualmente, 37% a mais em comparação ao mesmo período de 2013. Isto sugere que, cada vez mais, softwares para a web estão sendo criados, e pelo fato de serem direcionados à internet é de praxe que esses softwares sejam acessíveis de qualquer lugar do mundo para que possam abranger o maior público possível. Porém, a internet é um amplo sistema de comunicação que conecta muitas redes de computadores, as quais dependem de ativos físicos para a retransmissão da informação através de ambientes suscetíveis à adversidades como, falta de energia elétrica, rompimento dos meios de transmissão e até mesmo catástrofes climáticas.

Considerando estas possíveis adversidades, pode-se considerar que alguns softwares fiquem inacessíveis representando perdas financeiras ou mesmo a insatisfação por parte de seus clientes, de modo que, faz-se necessário o monitoramento destes softwares em períodos de tempo regulares para garantir que eles não ficarão inacessíveis e não sofrerão alterações indesejadas, como afirma Sloman (1994) apud Scherer (2012) p. 27:

[1] Netcraft é uma empresa de serviços de Internet que oferece análise de quota de mercado para mercados de hospedagem de sítios e de servidores web, incluindo detecção do sistema operacional e do servidor web.

> Gerenciar um sistema consiste em supervisionar e controlar seu funcionamento para que ele satisfaça aos requisitos tanto do seus usuários quanto do seus proprietários.

Assim sendo, este trabalho visou a criação de uma ferramenta para o monitoramento de softwares web, que permite a verificação da disponibilidade e integridade, através de análises em períodos de tempo regulares, a fim de identificar anormalidades que possam ser inferidas a eles. Para isto, foram desenvolvidos códigos de monitoramento que trabalham com o intuito de coletar informações como, tempo de resposta ou modificações nos softwares usando os protocolos utilizados atualmente na web.

2. Referencial Teórico

Nesta seção serão abordados conceitos relacionados ao grupo de ferramentas que serão desenvolvidas, permitindo assim, sua melhor compreensão.

2.1. A Internet e suas Redes de Computadores

A Internet é um sistema global de redes de computadores interligadas que utilizam um conjunto de regras que governam a comunicação dos dados, controlando o que é comunicado, quando e como são realizadas tais comunicações; essas regras são chamadas de protocolos [Forouzan e Fegan 2008].

A conexão de uma máquina à internet, caracteriza-se por ela estar conectada a uma rede, ser capaz de executar uma pilha[2] de protocolos do modelo TCP/IP (Seção 2.1.2) e poder enviar dados através do Protocolo de Internet (IP, do inglês, *Internet Protocol*) para outras máquinas na rede [Tanenbaum 2003].

A maioria das redes de computadores aceitam hierarquias de protocolos divididas em camadas, sendo que, cada uma fornece serviços às camadas superiores e as mantém isoladas dos detalhes dos protocolos usados nas camadas inferiores. Segundo Tanenbaum (2003), geralmente as pilhas de protocolos se baseiam no modelo OSI (*Open Systems Interconnection*) ou no modelo TCP/IP (Protocolo de controle de transmissão e protocolo de internet, do inglês, *Transmission Control Protocol/Internet Protocol*).

O modelo OSI se baseia em uma proposta desenvolvida pela ISO (*International Standarts Organization*) para padronização da interconexão entre sistemas, tendo como ponto chave sete camadas, onde cada camada executa uma função bem definida e se comunica com suas camadas adjacentes através de interfaces. Na Figura 1 é possível visualizar como essas camadas estão dispostas no modelo.

Já o Modelo TCP/IP tem como característica mais importante seus protocolos, tanto que o próprio nome do referido modelo se dá por causa de seus dois principais, o TCP e o IP.

[2] Uma pilha de protocolos refere-se a um conjunto de protocolos dispostos em camadas empilhadas.

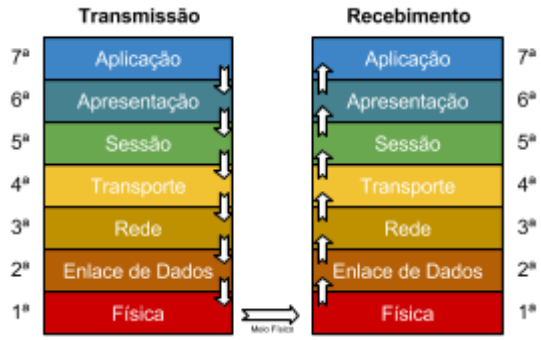

Figura 1 - Disposição das camadas do Modelo OSI e seu fluxo de informação [Adaptado de Forouzan e Fegan 2008].

Os modelos de referência OSI e TCP/IP apresentam muitos pontos em comum pois, ambos se baseiam em uma pilha de protocolos independentes[3] e suas camadas possuem características semelhantes, porém, há uma diferença significativa entre os dois modelos que é o número de camadas, já que, o modelo OSI possui sete camadas e o TCP/IP possui apenas quatro. Tais modelos são comparados na Figura 2.

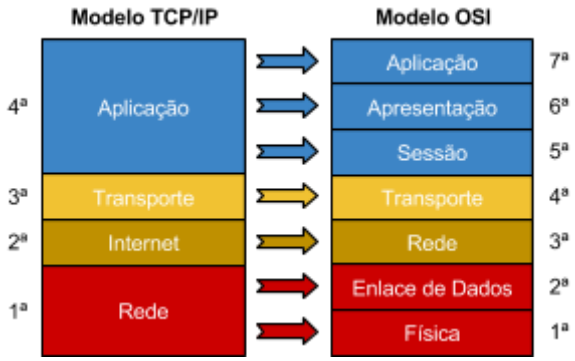

Figura 2 - Comparação entre as camadas do Modelo TCP/IP e o Modelo OSI [Adaptado de Forouzan e Fegan 2008].

2.2. Protocolos TCP/IP

O protocolo IP é o mecanismo de transmissão de dados usado pelos protocolos TCP/IP, o qual foi projetado com o objetivo de estabelecer a ligação entre redes interconectadas. Assim sendo, a função deste protocolo é fornecer a melhor forma de transportar datagramas[4] do local de origem para o destino, independentemente desses estarem na mesma rede ou em redes intermediárias. Porém, o IP não é um protocolo confiável, já que, não há garantia que os datagramas serão entregues corretamente ou mesmo que serão entregues em sua devida ordem, também não há nenhum mecanismo de detecção caso tais erros de transmissão ocorram. Todavia, isso não é considerado um problema do protocolo pois, desta forma, ele oferece as condições minimas necessárias para comunicação, proporcionando uma alta eficiência na transmissão dos datagramas [Forouzan e Fegan 2008].

---

[3] Apesar dos modelos possuírem pontos em comuns os protocolos são completamente diferentes.

[4] Um datagrama é uma unidade de transferência básica associada a uma rede de comutação de pacotes em que a entrega, hora de chegada e a ordem não são garantidos [Kurose e Ross 2010].

Para poder fazer o transporte de datagramas, o protocolo IP exige um endereço de rede chamado de endereço IP para toda e qualquer máquina que pertença a uma rede, de modo análogo ao sistema telefônico, onde cada assinante tem um número de telefone exclusivo no mundo. Entretanto, o endereço IP possui duas versões atualmente em uso, a versão 4, composta por um número de 32 bits, comumente denominada IPv4, e a versão 6, composta por um número de 128 bits, comumente denominada de IPv6. Cada uma destas versões possui uma notação própria, sendo que, o IPv4 possui uma notação decimal que é dividida em octetos (8 bits), formando 4 bytes de informação separadas por ponto (“.”). Assim sendo, um exemplo de endereço IPv4 poderia ser descrito como, “123.123.123.123”, onde cada número decimal deve estar compreendido entre 0 e 255. Já a notação do IPv6 se dá através de oito grupos de quatro dígitos hexadecimais (16 bits) separados pelo sinal de dois pontos (“:”). Caso o grupo seja composto somente de números zero, o mesmo pode ser totalmente omitido, de modo que, um endereço IPv6 poderia ser descrito como, “2001:0db8:85a3:0000:0000:0000:0000:7344” ou mesmo como, “2001:0db8:85a3::7344”.

Tendo em vista que o protocolo IP não é confiável, Tanenbaum (2003) menciona o protocolo TCP (que opera na camada de transporte), o qual foi projetado para complementar o protocolo IP (que opera na camada de rede) ao oferecer-lhe uma entrega confiável e em sequência, permitindo assim, conexões *full-duplex* e ponto a ponto, ou seja, conexões em que o tráfego ocorra em ambas as direções ao mesmo tempo, e que cada conexão possua somente uma origem e um destino.

Uma conexão TCP é obtida quando, tanto o transmissor quanto o receptor criam pontos extremos de conexões chamados soquetes. Cada soquete possui um endereço único que é composto do endereço IP da máquina e de um número decimal de 16 bits, compreendido entre 0 e 65535 chamado porta. Tais conexões de soquetes são estabelecidas por meio de um processo que ocorre em três passos, conhecido por aperto de mãos de três vias (do inglês, *three handshake*), que funciona com um soquete “B” aguardando passivamente por uma conexão de entrada, ou seja, em estado de espera (do inglês, *LISTEN*), enquanto um soquete “A”, solicita a conexão (do inglês, *CONNECT*) especificando o endereço do soquete “B” ao qual deseja se conectar. Para que isso ocorra, o soquete “A” envia um datagrama TCP com o bit Sincronizar (*SYN,* do inglês, *Synchronize)* ativado e um bit Confirmação (*ACK,* do inglês, *Acknowledgment)* desativado, e aguarda a resposta. Quando a solicitação chega ao soquete “B”, o mesmo retorna uma resposta contendo o bit *SYN* ativado e o bit *ACK* ativado, fazendo com que o solicitante envie outro bit *ACK* ativado para confirmar que a conexão foi estabelecida. Caso o soquete “B” recuse a conexão, será retornado uma resposta contendo o bit Encerrar (*RST,* do inglês, *reset)* ativado. Já o encerramento da conexão se dá através do envio do bit Finalizar (*FIN*, do inglês, *Finalize)* pelos dois lados, seguido de sua confirmação pelo bit *ACK* ativado. Todo este processo é ilustrado na Figura 3 [Tanenbaum 2003].

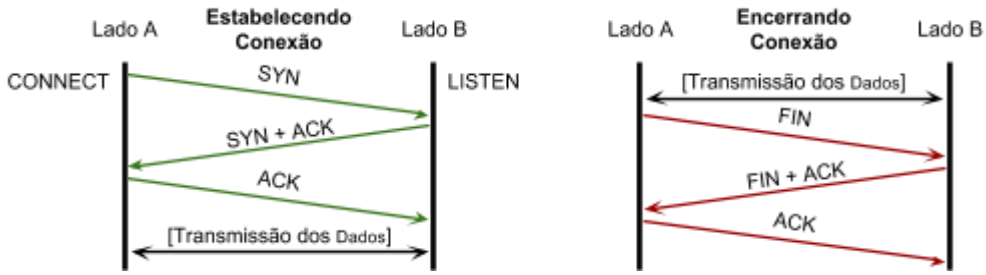

Figura 3 - Ciclo de vida ideal do estabelecimento de um conexão TCP até sua finalização por ambos os lados [Adaptado de Tanenbaum 2003].

2.2.1. Protocolo ICMP

Como o protocolo IP não possui um mecanismo de relatório de erros, o Protocolo de Mensagens de Controle da Internet (ICMP, do inglês, *Internet Control Message Protocol*) foi criado para auxiliá-lo, trabalhando na camada de Rede com o objetivo de fornecer relatórios por meio de mensagens de controle preestabelecidas [Forouzan e Fegan 2008].

A finalidade dessas mensagens é fornecer *feedback* sobre problemas no ambiente de comunicação através de 11 mensagens de controle, as quais estão definidas em um documento de padronização da internet denominado RFC792 (1981), onde, as únicas importantes no desenvolvimento deste trabalho foram as mensagens de número 8 (*ECHO*), 0 (*ECHO REPLY*), 13 (*TIMESTAMP*) e 14 (*TIMESTAMP REPLY*), que segundo Tanenbaum (2003), são usadas para verificar se um determinado destino está ativo e acessível.

Estas mensagens, em especifico, funcionam como uma simples pergunta e resposta, onde uma máquina "A" envia um datagrama ECHO (pergunta) para uma máquina "B", e esta ao recebe-lo devolve uma mensagem ECHO REPLY (resposta); as mensagens TIMESTAMP e TIMESTAMP REPLY funcionam analogamente, apenas com a adição do horário corrente aos datagramas [Tanenbaum 2003].

2.2.2. Protocolo DNS

O protocolo conhecido por Sistema de Nomes de Domínios[5] (DNS, do inglês *Domain Name System*) é um protocolo de suporte[6] que trabalha na camada de Aplicação do modelo TCP/IP, sendo responsável pela nomenclatura na Internet. Tal nomenclatura é necessária pois, os endereços numéricos descritos no protocolo IP são difíceis de memorizar e causam uma dependência indesejável, por exemplo, caso a máquina detentora de determinado endereço IP mude seu endereço, todas as referências a esta máquina terão que ser atualizadas para o seu novo endereço. Para resolver esse problema, o DNS introduziu o conceito de nomes para as máquinas, desacoplando assim, os endereços IP dos endereços das máquinas, ou seja, a principal função do DNS é traduzir o nome de uma máquina em seu respectivo endereço IP e vice-versa [Kurose

[5] Domínio é um nome que serve para localizar e identificar conjuntos de computadores na net. O nome de domínio foi concebido com o objetivo de facilitar a memorização dos endereços de computadores na Internet, pois, sem ele seria necessário memorizar uma sequência grande de números [CGI 2014].

[6] Protocolo usado indiretamente por outros protocolos ou aplicações, porém é de fundamental importância para o funcionamento da Internet.

e Ross 2010].

O DNS trabalha com um sistema hierárquico de atribuição de nomes, onde cada ramificação dá origem a um domínio que, por sua vez, é particionado em subdomínios que também são particionados e assim por diante, de modo que, atualmente, a internet é dividida em 512 domínios de topo[7] [ICANN 2014]. A exemplo disso é possível citar o domínio de topo de terminação BR, o qual é atribuído a República Federativa do Brasil, e como subdomínio UNIFRA, atribuído ao Centro Universitário Franciscano, uma instituição privada de ensino superior localizada no Brasil, e ainda dividí-lo no subdomínio WWW[8], que é atribuído a máquina servidora de conteúdo web dentro da instituição. Com isso, cria-se uma hierarquia, como visto na Figura 4.

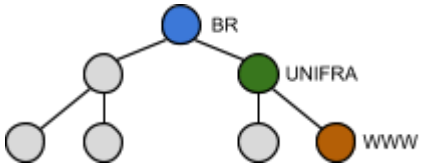

Figura 4 - Exemplo de uma hierarquia DNS. Onde pode ser visualizada a formação do domínio www.unifra.br.

Para gerar um nome único que identifique a máquina WWW os domínios e subdomínios são concatenados em ordem inversa a sua hierarquia, utilizando um ponto "." acrescido de outro ponto para identificar o topo, com isso é obtido o nome "WWW.UNIFRA.BR.", denominado de Nome de Domínio Totalmente Qualificado (FQDN, do inglês, *Fully Qualified Domain Name*).

Quando uma resolução de nome é solicitada ao servidor de DNS, ele retorna um conjunto de registros de recursos, sendo que, cada registro de recurso é representado - em modo texto - por um conjunto de cinco campos, contendo: *NAME*, *TTL*, *CLASS*, *TYPE* e *VALUE*.

- *NAME*: É o FQDN ao qual esse recurso se aplica.
- TTL (Tempo para Viver, do inglês, *Time To Live*): Fornece uma indicação de estabilidade do recurso, ou seja, com que frequência em segundos ele é alterado, por exemplo, caso esse valor seja baixo, em torno de 3600 segundos, indica que este recurso é instável.
- CLASS: É a classe ao qual esse recurso pertence, e sempre é representado por "IN" (do inglês, *Internet*) em registros relacionados à internet.
- TYPE: É o tipo de recurso que será descrito no campo VALUE. Existem vários tipos padronizados para uso na internet e cada um deles tem uma sintaxe própria definida em documentos RFCs.
- VALUE: É a informação própriamente dita do recurso, e depende diretamente do tipo associado.

[7] O domínio de topo (TLD, do inglês *top-level domain*) é um dos componentes dos endereços de Internet. Cada nome de domínio na Internet consiste de alguns nomes separados por pontos, e o último desses nomes é o domínio de topo.

[8] O domínio WWW é um prefixo comumente usado na web, porém seu uso não é obrigatório, e fica a critério de quem está definindo o nome para suas máquinas na internet.

2.2.3. Protocolo HTTP e a World Wide Web

Na década de 90, o físico Tim Berners-Lee formalizou a Teia Mundial de computadores (WWW, do inglês, World Wide Web), comumente chamada somente de web, que trata-se de uma aplicação da Internet que utiliza o conceito de cliente-servidor distribuído, no qual um usuário usando um computador que disponha de um software denominado Navegador (cliente) pode acessar um amplo conjunto de documentos de hipermídia[9], frequentemente designados de sítios ou de páginas web, que estão armazenados em computadores espalhados pela Internet denominados servidores [Tanenbaum 2003].

Cada página web possui um endereço único na internet chamado de Localizador Padrão de Recursos (URL, do inglês, *Uniform Resource Locator*), cuja criação tem como objetivo responder três perguntas: como determinada página pode ser acessada?, em que lugar ela está localizada? e qual é o caminho para se chegar a essa página, obtendo como resultado a seguinte sintaxe: protocolo://sitio:porta/caminho, onde o protocolo define como a página será obtida, por exemplo, http. Já o sítio é o lugar em que a página pode ser obtida, representado pelo nome de domínio ou do endereço IP da máquina servidora. Enquanto que a porta complementa o sítio para formar o endereço do soquete a ser acessado pelo Navegador. Por fim, tem-se o caminho usado para especificar o local da página dentro do sítio. Dentre esses, os únicos parâmetros mandatórios são o protocolo e o sítio, sendo os outros opcionais, e quando omitidos valores padrões são usados [Forouzan e Fegan 2008]. Um exemplo de URL pode ser dada por: “http://www.unifra.br:80/novo/site/default.aspx”, ou seja, através do protocolo http, busca-se o sítio www.unifra.br, que encontra-se disponível na porta 80, onde possui uma página web no caminho /novo/site/default.aspx. Tal endereço poderia ser escrito omitindo-se a porta e o caminho já que essas informações são de fato padrões para este servidor web, gerando assim a URL: “http://www.unifra.br”.

Para que um cliente possa recuperar uma página web armazenada em um servidor, é necessário que tanto o cliente quanto o servidor usem o mesmo protocolo de comunicação, que no caso da web é o Protocolo de Transferência de Hipertexto (HTTP, do inglês, *Hypertext Transfer Protocol*) [Forouzan e Fegan 2008].

O HTTP funciona com uma combinação de transferência de arquivos e mensagens, onde cada transação utiliza apenas uma conexão através de soquetes TCP e, é composta por um pedido e uma resposta. Uma mensagem de pedido possui: uma linha de pedido, um cabeçalho e, opcionalmente, um conteúdo ou arquivo. Já uma mensagem de resposta consiste em: uma linha de status, um cabeçalho e normalmente um conteúdo ou arquivo.

Em uma mensagem de pedido, a “Linha de Pedido” é composta pelo tipo do pedido, a URL e a versão do protocolo usado, sendo que, o único tipo relevante para este trabalho é o denominado *GET*, que é responsável por solicitar um documento do servidor. Já a versão do protocolo é enviada a fim de garantir que, tanto o cliente como o servidor se comuniquem utilizando a mesma versão.

---

[9] Hipermídia é a reunião de várias mídias num formato computacional, suportado por sistemas eletrônicos de comunicação [Laufer e Scavetta 1993].

Em uma mensagem de resposta, a “Linha de Status” é composta por: uma versão do protocolo e um código de status seguido de uma frase de status, onde cada código é composto por um número de três dígitos e dividido em cinco intervalos definidos pelo primeiro dígito:

1xx. São reservados para fins informativos;
2xx. Indicam que o pedido foi bem-sucedido;
3xx. Informam ao cliente que ele deve refazer a solicitação em outra URL;
4xx. Avisam que o cliente fez uma solicitação de forma incorreta;
5xx. Avisam que ocorreu um erro no servidor ao processar a solicitação.

Na sequência de uma mensagem, seja ela de pedido ou resposta, tem-se o cabeçalho, o qual contém informações adicionais sobre a mensagem que está sendo transmitida e pode conter uma ou várias linhas, onde cada linha é composta por: um Nome seguido de dois-pontos “:” , um espaço em branco e um Valor, formando a seguinte sintaxe “nome: valor”, assim é possível anexar qualquer informação adicional a uma mensagem HTTP. Porém, a RFC2616 (1999) especifica uma série de cabeçalhos que podem ser usados conforme desejado, mas isso não impede a adição de novas linhas personalizadas. Por exemplo, um cliente ao solicitar uma página web pode especificar em qual idioma ela deve ser entregue, e o servidor, ao entregar a página, pode especificar outra linha no cabeçalho informando quando essa determinada página foi criada, apresentando assim a configuração fictícia apresentada na Figura 5.

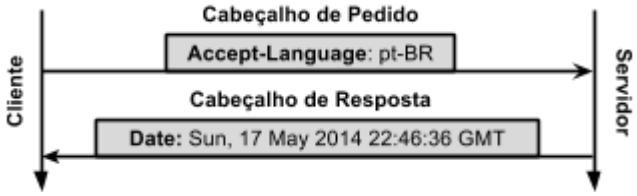

Figura 5 - Exemplo de fragmentos dos cabeçalhos em uma Transação HTTP obtidas de uma solicitação ao sítio unifra.br.

E por último, tem-se o conteúdo ou arquivo, que no caso de uma requisição do tipo *GET*, poderá estar presente somente na mensagem de resposta e, normalmente, é o conteúdo hipertexto que será processado e apresentado pelo navegador web ao usuário que está utilizando-o.

Apesar da complexidade existente em um cliente navegador para acessar a web, atualmente tal tarefa é trivial pois, a maioria das linguagens de programação existentes já estão preparadas para trabalhar com os protocolos nela utilizados, e as que não estão preparadas, em sua maioria, possuem módulos ou implementações para tal utilização, abstraindo assim, a complexidade dos protocolos aqui descritos.

2.3. Linguagem de Programação Python

Python é uma linguagem de programação de alto nível (VHLL, do inglês, *Very High Level Language*) pois, assemelha-se muito à linguagem humana. Possui uma sintaxe moderna que permite maior foco no problema a ser resolvido e suporta vários paradigmas de programação como, o estruturado, o orientado a objetos e também a alguns elementos da programação funcional. Porém, nenhuma destas escolhas é imposta

ao programador, sendo possível utilizar e mesclar os paradigmas de acordo com as necessidades de quem a utiliza. Outro fator relevante é o fato dela ser interpretada, ou seja, não há necessidade de compilar o código escrito em código de máquina, pois este processo é feito automaticamente durante a execução do mesmo, fazendo com que o desenvolvedor que a emprega seja mais produtivo do que usando uma linguagem compilada [Python 2014].

Sendo uma linguagem de programação multiparadigma e modular, Python é considerada uma linguagem de uso geral e pode ser empregada em vários tipos de problemas, entre eles, os relacionados à internet [Beazley e Jones 2013], pois em suas bibliotecas padrões estão incluídos módulos para o processamento de textos usando expressões regulares através do módulo *re*, analise de URLs através do módulo *urllib*, comunicação por soquetes TCP através do módulo *socket*, manipulação de horários e datas através do módulo *datetime* e chamadas a processos de sistema usando o módulo *subprocess*. Todos estes modulos utilizados neste trabalho, assim como outras bibliotecas desenvolvidas por terceiros, como por exemplo, para manipulação do protocolo DNS através da biblioteca *py3dns*, conexão a banco de dados usando *mongoengine*, controle da distribuição de tarefas através do *celery* e interface de comunicação com o terminal de comandos através do *click*.

2.4. Metodologia de Gerenciamento de Projetos - DSDM

Segundo Teixeira et al. (2005) o projeto de um software - seja uma aplicação ou ferramenta - deve ser regido por uma metodologia para mitigar possíveis erros durante o seu desenvolvimento. A fim de orquestrar um projeto ele propõe o uso da Metodologia de Desenvolvimento de Sistemas Dinâmicos (DSDM, do inglês, *Dynamic Systems Development Method*), definida como uma Metodologia Ágil[10] de desenvolvimento de software iterativo e incremental que enfatiza o envolvimento constante do usuário.

Seu objetivo é entregar softwares com tempo e custos apertados, através do controle e ajuste de requisitos ao longo do desenvolvimento de três fases, sendo que a primeira - o pré-projeto - consiste em validar se a metodologia é a mais adequada ao projeto, bem como definir um orçamento e um contrato entre as partes interessadas. Já a segunda fase é composta pelo ciclo de vida do projeto e a terceira fase é o pós-projeto que envolve a manutenção do sistema após a sua entrega.

O ciclo de vida do projeto é a fase mais extensa da metodologia, portanto é dividida em quatro estágios, onde o primeiro é sequencial e os outros três são iterativos:

- Análise: é um processo único onde é feito um estudo de viabilidade e um estudo do negócio ao qual o projeto será aplicado.
- Iteração do Modelo Funcional: é o processo de desenvolvimento de um protótipo utilizável (funcional) que engloba a escolha de que parte do sistema será desenvolvido, a criação de um cronograma e o desenvolvimento própriamente dito.
- Iteração de Desenho e Construção: é o processo de criação do software

[10] Metodologias Ágeis são um grupo de metodologias de desenvolvimento de software que providenciam uma estrutura conceitual para reger projetos de engenharia de software, onde são valorizados os indivíduos que o compoêm, o software em funcionamento, a colaboração com o cliente e a resposta a mudanças [Teixeira et al. 2005].

com base no Modelo Funcional.

- Implantação: é o processo de configuração do software e treinamento de usuários sobre as novas funcionalidades implementadas. Após esse estágio volta-se para o Modelo Funcional para o desenvolvimento de uma nova funcionalidade.

3. Trabalhos Correlatos

Neste capítulo são abordados alguns trabalhos disponíveis na literatura relacionada a serviços de monitoramento. Porém, há um foco comum entre os trabalhos relacionados, o fato de todos eles apresentarem uma configuração semelhante, onde é necessário uma prévia configuração do alvo a ser monitorado.

3.1. Análise de Ferramentas de Monitoração de Código Aberto

De acordo com Mohr (2012), existe a necessidade do uso de ferramentas de monitoramento para detectar problemas antes que os usuários percebam que existe algo errado acontecendo. Para isto, o autor avalia e compara três ferramentas de monitoração, desenvolvidas em código aberto, disponíveis atualmente no mercado: Nagios, Zenoss e Zabbix, as quais, segundo o autor, foram escolhidas devido a facilidade de mudança que elas apresentam, por serem de código aberto, e ainda por possuírem algumas características particulares que as diferem, sendo que, o Nagios foi escolhido por ser uma ferramenta voltada a usuários mais avançados, já o Zenoss é voltado para a interface gráfica e compatibilidade com *plugins*[11] já existentes, e o Zabbix é uma mescla dessas duas ferramentas que, conforme o autor, possui o potencial para ser a mais completa entre as três.

Porém, em todas as ferramentas avaliadas o autor aborda os protocolos de monitoramento desenhados para trabalhar de forma ativa, assim sendo, cada monitor possui um gerente que está ligado a um ou a vários agentes monitores instalados diretamente nas máquinas monitoradas. Isso propícia um monitoramento mais detalhado a respeito do que está sendo monitorado, porém, faz com que este tipo de monitoramento seja intrusivo e exija uma prévia configuração dos clientes a serem monitorados.

3.2. Service Level Agreement Monitor (SALMon)

Ameller e Franch (2008) descrevem a criação de uma ferramenta chamada SALMon que utiliza uma técnica de monitoramento para fornecer maior autonomia na Qualidade de Serviços (QoS, do inglês, Quality of Service), a qual é necessária para detectar violações no Acordo de Nível de Serviço[12] (SLA, do inglês, Service Level Agreement) em aplicações web usando a tecnologia de *Web-Services,* baseada em uma Arquitetura Orientada a Serviços (SOA, do inglês, Service-Oriented Architecture).

Para isso, foi desenvolvido três tipos de serviços, o Monitor, o Analisador e o

[11] Um plugin normalmente é um pequeno programa de computador usado para adicionar funções a programas maiores, provendo alguma funcionalidade especial ou específica.

[12] SLA é um documento exigido na maioria dos contratos de serviços de TI, além de ser um documento da ITIL que sustenta a ISO/IEC 20000 sobre gerenciamento de qualidade de serviços de TI, é nele que estarão definidos, aceitos e formalizados os níveis de serviço esperados pelo cliente de TI.

Decisor. O serviço Monitor é composto por instrumentos de medição, os quais são chamados iterativamente pelo Monitor em intervalos de tempo preestabelecidos, e suas respostas são armazenadas em um banco de dados para posteriormente serem analisadas pelo Analisador e, caso alguma alteração na SLA seja encontrada, o Decisor será acionado para decidir qual a melhor tarefa preestabelecida a ser utilizada para corrigir a possível falha que ocasionou as alterações encontradas pelo Analisador.

Os autores afirmam que, tal técnica pode ser empregada para monitorar outros tipos de serviços como, banco de dados.Todavia, tal modelo de monitoramento não permite sua utilização em arquiteturas que não sejam SOA, ou seja, não seria possível aplicar a ferramenta SALMon em uma arquitetura web convencional, além disso, assim como o trabalho descrito por Mohr (2012), o SALMon utiliza uma técnica ativa de monitoramento.

3.3 Cloud Service Monitoring System Based on SLA

Assim como Ameller e Franch (2008), Liu e Xu (2013) focam seus esforços no monitoramento de SLA através de uma interface de *Web-Services,* desenvolvida usando a Linguagem de Descrição para Serviços Web (WSDL, do inglês, *Web Services Description Language*), obtendo assim, uma ferramenta robusta e flexível para monitorar e gerenciar SLAs.

No entanto, devido o foco de sua ferramenta voltada para o monitoramento de SLA, Liu e Xu (2013) tornam o serviço oferecido pela ferramenta incapaz de ser usado para simples monitoramento de uma aplicação web sem implicações em um contrato de serviço.

3.4. Considerações

Apesar dos softwares e ferramentas de monitoramento apresentados por Mohr (2012), Ameller e Franch (2008) e Liu e Xu (2013) empregarem algumas das técnicas que serão apresentadas neste trabalho, o resultado do monitoramento não é direcionado a supervisão passiva de sítios web, mas sim, a supervisão ativa de servidores e serviços. Isso acarreta em uma pré-configuração dos alvos a serem monitorados, o qual não é objetivo deste trabalho, e devido a natureza dos softwares de monitoramento apresentados, existe a necessidade de que eles sejam cuidadosamente preparados para o monitoramento exigindo uma compreensão de suas configurações específicas, que podem abranger um entendimento avançado por parte de quem os opera.

Entretanto, assim como as ferramentas descritas por Ameller e Franch (2008) e Liu e Xu (2013), este trabalho utiliza um Monitor composto por instrumentos de medição, os quais poderão ser chamados repetidamente em intervalos de tempo preestabelecidos, sendo que, as respostas obtidas dos instrumentos de medição também são armazenadas em um banco de dados. Contudo, o trabalho aqui descrito analisa as respostas e compara com dados históricos dos instrumentos de medição no momento em que elas são geradas, e caso alguma alteração seja encontrada uma mensagem de alerta é emitida.

4. Metodologia

Este trabalho fez uso da metodologia DSDM durante o desenvolvimento da ferramenta

para monitoramento de softwares web, que se propõe a coletar dados dos alvos monitorados através de requisições, usando os protocolos da web para posteriormente analisar suas respostas.

Para proporcionar tal forma de monitoramento, foram desenvolvidas as funções de: Monitor que irá chamar, iterativamente, alguns Instrumentos de Monitoramento (IMs) - conforme informações provindas de um banco de dados - e um Analisador, o qual irá processar as respostas dos IMs a fim de encontrar possíveis anomalias ou informações que possam caracterizar um problema de disponibilidade e/ou integridade, gerando assim, uma mensagem de alerta que, por sua vez, será gravada em um banco de dados para posterior conferência. Tal estrutura pode ser vista na Figura 6.

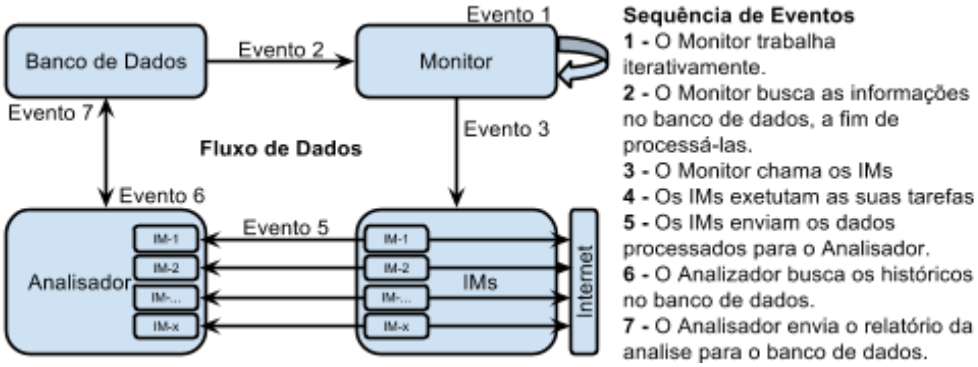

Figura 6 - Fluxo de informação da ferramenta de monitoramento web.

4.1 - Monitor

A tarefa do Monitor é buscar iterativamente as informações no banco de dados a fim de descobrir quais IMs devem ser chamados e quais parâmetros serão passados para eles. Assim, cada IM irá executar uma tarefa específica a fim de coletar informações referente ao alvo monitorado e, posteriormente, devolver estas informações ao Analisador que, por sua vez, armazenará seus resultados no banco de dados.

4.1.1. IM de Disponibilidade do Servidor - ICMP

A tarefa de monitoramento da disponibilidade do servidor foi realizada através do protocolo ICMP, executando quatro requisições ao alvo monitorado a fim de obter como resposta a média do tempo de conexão (MT1). Tal monitoramento foi possibilitado pelo utilitário de rede "ping", um comando disponível na maioria dos sistemas operacionais.

De posse da MT1, o IM retorna uma mensagem para seu respectivo Analisador contendo as informações de sua execução, dentre elas: uma mensagem, um código numérico e o próprio MT1, necessárias para validar a execução do comando, bem como calcular a média de tempo de conexão das ultimas 120 iterações (MT2). Assim, se ao realizar a comparação "(MT2 x 2) < MT1" uma resposta verdadeira for obtida, um alerta do tipo Aviso (do inglês, *Warning*) é gerado e gravado no banco de dados. Mas antes que isto ocorra o código numérico é analisado, e ao ser diferente de zero representa um alerta do tipo Perigo (do inglês, *Danger*) e é descrito com mais detalhes na mensagem anexa, que a exemplo pode significar que o alvo monitorado está indisponível ou mesmo que o comando falhou.

4.1.2. IM de Disponibilidade do Serviço - TCP

O IM de disponibilidade do serviço assemelha-se muito com o IM de Disponibilidade do Servidor (ICMP), porém, ao invés de utilizar o utilitário "ping" ele utiliza o módulo *socket* da linguagem Python para realizar a conexão e obter um tempo de resposta.

Tal tempo de resposta é enviado ao seu respectivo Analisador que, assim como o IM anteriormente descrito, valida a execução do comando e calcula a média das ultimas iterações a fim de obter os seus respectivos alertas.

4.1.3. IM de Disponibilidade e Integridade do Servidor de Nomes - DNS

Este IM é responsável por coletar um conjunto de registros de recursos, assim como seu tempo de resposta, referentes ao protocolo DNS do alvo monitorado, fazendo uso do modulo *py3dns*.

Essas informações são enviadas ao Analisador, que valida os dados (assim como os IMs anteriores) e compara o conjunto de registros de recursos atual com o ultimo conjunto obtido, o qual, em caso de divergência, grava um aviso do tipo "Perigo" no banco de dados.

4.1.4. IM de Disponibilidade e Integridade do Conteúdo - HTTP

Devido a complexidade envolvida na utilização do protocolo HTTP, foi utilizado para a criação deste IM um programa para simulação de um Navegador Web real, denominado *phantomjs*, que tem por objetivo simular um utilizador humano para obter um conjunto de informações a respeito da página web monitorada e sua comunicação.

Esse navegador possui uma Interface de Programação de Aplicativos (API, do inglês, *Application Programming Interface*) capaz de interagir com outros aplicativos, de modo que, utilizando a linguagem de progração Python através do módulo *subprocess*, foi possível repassar os comandos para que ele retornasse um conjunto de informações composto por um código numérico que identifica o sucesso da execução, uma mensagem informativa sobre a execução do programa, o código HTML da requisição, assim como o estado de todas as requisições encadeadas por esse código.

Através dessas informações o Analisador é capaz de identificar se o alvo monitorado encontra-se indisponível, gerando, portanto, um alerta do tipo Perigo ou mesmo identificar se o tempo de resposta é superior a média das ultimas requisições, lançando assim um alerta do tipo Aviso.

4.2. Distribuição das Tarefas

O processamento por parte dos IMs constatou-se demorado e consumiu muitos recursos computacionais, de forma que fez-se necessário a distribuição dos IMs para não sobrecarregar uma única máquina. Também, a distribuição possibilitou espalhar geográficamente os IMs, aumentando a confiabilidade da ferramenta, já que as redes de computadores ao qual a ferramenta está submetida é a mesma que ela monitora, ou seja, aumentando o número de redes utilizadas ao redor do mundo é possível aumentar sua confiabilidade.

A fim de suprir essa necessidade de distribuição dos IMs, seja geograficamente

ou para fins de sobrecarga, foi utilizado um aplicativo para Filas de Tarefas Distribuídas (do inglês, Distributed Task Queue) denominado *Celery* em conjunto com o servidor de mensagens *RabbitMQ*. O aplicativo *Celery* trabalha com um conjunto de três utilitários: o Cliente (incorporado pela funcionalidade de Monitor), os Trabalhadores (representados por um ou vários IMs) e o Servidor (representado pelo aplicativo *RabbitMQ)*.

Assim, foi possível definir que cada IM será a tarefa de um ou vários Trabalhadores, a fim de serem chamadas pelo Monitor usando o Cliente da fila, fazendo com que o Servidor escolha qual Trabalhador deverá executar a tarefa solicitada. Tal estrutura é apresentada na Figura 7.

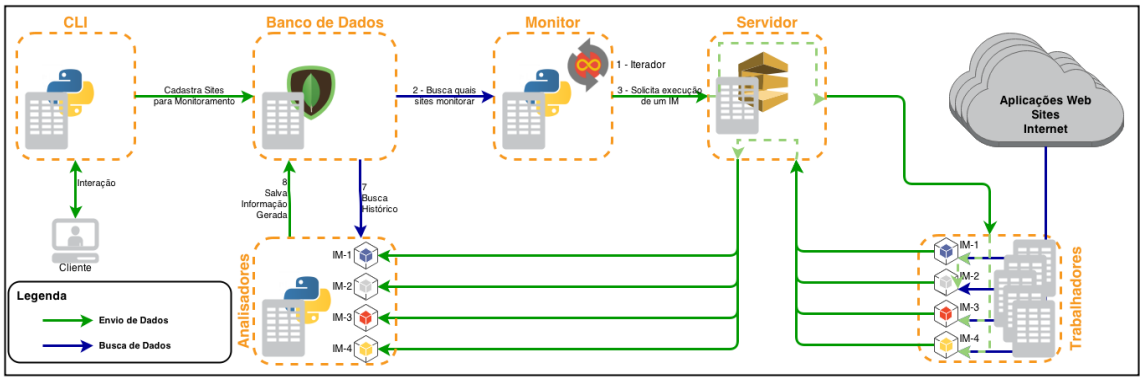

Figura 7 - Estrutura da ferramenta, englobando a interface (CLI), o Monitor, os Instrumentos de Monitoramento, os Analisadores, o Banco de Dados e como eles se comunicam.

5. Estudo de Caso

A fim de validar a utilização da ferramenta descrita neste trabalho foram realizados testes utilizando todos os IMs tendo como alvo os sítios bolzan.io localizado nos Estados Unidos da América e globo.com localizado no Brasil, ambos disponíveis publicamente na internet, sendo o primeiro de conteúdo estático, portando não apresentando alterações com o decorrer do tempo e o segundo de conteúdo dinâmico, sofrendo alterações constantes em sua composição ao longo do tempo.

Todos os testes foram aplicados utilizando uma infraestrutura computacional virtualizada em um ambiente GNU/Linux composta por uma máquina contendo o Monitor, o Analisador, o banco de dados e o Servidor das filas e outra máquina contendo os Trabalhadores.

Os IMs foram cadastrados para executar em um intervalo de tempo de um minuto, visto que este é o menor tempo possível para a execução dos IMs, caracterizando a máxima capacidade da ferramenta no que tange a intervalo de execução.

No decorrer do processo de monitoramento a ferramenta foi submetida a uma falha na rede de internet, simulando assim uma situação de indisponibilidade nos sítios monitorados.

6. Resultados

O monitoramento através dos IMs "ICMP" e "TCP", tendo como alvo o sítio bolzan.io, obtiveram uma média aproximada de tempo de resposta de 200 milisegundos, enquanto que para o sítio globo.com obtiveram aproximadamente 25 milisegundos; isto ocorreu porque o alvo bolzan.io está localizado a uma maior distância que o alvo globo.com. Ao ser submetido a uma falha de rede, a ferramenta reportou imediatamente o erro por meio de um aviso do tipo *Danger*, juntamente com a mensagem "*Hostname could not be resolved*", que significa que o alvo monitorado não está acessível, caracterizando uma indisponibilidade dos sítios analisados.

O monitoramento através do IM "DNS" agiu analogamente aos IMs anteriores, obtendo tempos de resposta semelhantes, e ao realizar suas analises, não encontrou nenhuma anomalia decorrente da alteração de DNS (durante um dia de monitoramento), porém, ao ser submetido à uma falha de rede, obteve como resposta um aviso contendo a mensagem "*Network is unreachable*", que em tradução livre significa que a rede está inacessível.

Já o monitoramento através do IM "HTTP" obteve tempos de resposta bem mais elevados, visto que este IM simula um navegador web real, portando, ele apresenta o tempo efetivo de resposta que este sítio demora para apresentar seu conteúdo para os usuários que o acessam através de navegadores comuns. Tais tempos de reposta foram de 4000 milisegundos (quatro segundos) para o sítio globo.com e 1000 milisegundos para o sítio bolzan.io, essa diferença se deve, principalmente, por causa de seus conteúdos, no qual o primeiro é dinâmico e o segundo estático. Ao ser submetido à falha de rede obteve as mesmas mensagens e avisos apresentados pelos IMs "ICMP" e "TCP".

Durante as situações de monitoramento realizadas, a ferramenta não apresentou falhas de execução ou instabilidades, reportando todos os resultados como o objetivado.

7. Conclusões

De acordo com os resultados apresentados, pode-se concluir que a ferramenta cumpre seus objetivos de forma satisfatória, demonstrando agilidade nas respostas e alta capacidade de monitoramento através da criação de múltiplas máquinas trabalhadoras, fazendo com que seja possível ampliar a capacidade da ferramenta simplesmente adicionando mais máquinas trabalhadoras, técnica essa chamada de escalabilidade horizontal.

Demonstrou-se também que, ao usar os IMs através das filas distribuídas, foi possível separar totalmente a máquina trabalhadora das demais, possibilitando uma distribuição geográfica da mesma.

Assim, em comparação as ferramentas apresentadas nos trabalhos relacionados, pode-se destacar o fato da ferramenta desenvolvida ter sido projetada para o monitoramento passivo de sítios web e a sua capacidade de ser distribuída.

A fim de dar seguimento nessa linha de pesquisa e trabalho, pode-se sugerir que sejam estudadas outras formas de monitoramento, assim como interfaces gráficas para

analise e exibição dos resultados fornecidos pela ferramenta, possibilitando, dessa forma, que usuários menos especializados possam utilizá-la.

Referências

Ameller, David; Franch, Xavier. (2008) “Service level agreement monitor (SALMon). In: Composition-Based Software Systems”. ICCBSS. Seventh International Conference on. IEEE. p. 224-227.

CGI (2014) “Domínios”, http://registro.br. Junho.

Forouzan, Behrouz A.; Fegan, Sophia Chung. (2008) “Protocolo TCP/IP”. 3. ed. São Paulo: Mcgraw-hill. 864 p.

ICANN (2014) “Root Zone Database”, http://www.iana.org/domains/root/db. Março.

Kurose, James F.; Ross, Keith W.. (2010) “Redes de computadores e a internet: Uma abordagem top-down”. 5. ed. [s. L.]: Pearson. 614 p.

Laufer, Roger; Scavetta, Domenico. (1993) “Texto, Hipertexto, Hipermedia”. Lisboa: Rés Formalpress. 160 p.

Liu, Xuan; Xu, Feng. (2013) “Cloud Service Monitoring System Based on SLA. In: Distributed Computing and Applications to Business”, Engineering & Science (DCABES), 12th International Symposium on. IEEE. p. 137-141.

Mohr, Rodrigo Fraga. (2012) “Análise de Ferramentas de Monitoração de Código Aberto”. 70 f. TCC (Graduação) - Curso de Ciência da Computação, Instituto de Informática, Universidade Federal Do Rio Grande Do Sul, Porto Alegre.

Netcraft (2014) (Inglaterra) (Org.). “January 2014 Web Server Survey”. http://goo.gl/w2fnqz. Março.

Python. (2014) “O que é Python?”. http://www.python.org.br. Maio.

Beazley, David. K. Jones, Brian. (2013) “Python Cookbook” O'Reilly Media, 706p.

Scherer, Pietro. (2012) “Monitoramento De Redes TCP/IP: Estudo De Caso Em Um Provedor De Internet”. 57 f. TCC (Graduação) - Curso de Gestão de Tecnologia da Informação, Centro de Ciência da Economia e Informática, Universidade da Região da Campanha, Caçapava do Sul, 2012. http://goo.gl/24iQtO. Março 2014.

Tanenbaum, Andrew S.. (2003) “Redes de computadores”. 4. ed. Rio de Janeiro: Elsevier. 945 p.

Teixeira, Daniel Dinis et al. (2005) “DSDM – Dynamic Systems Development Methodology” 14 f. Faculdade de Engenharia da Universidade do Porto.

The Internet Engineering Task Force. (1981) “RFC 792: Internet Protocol - DARPA Internet Program Protocol Specification”. EUA: IETF, Setembro 1981. 20 p. http://www.ietf.org/rfc/rfc792.txt. Março 2014.

The Internet Engineering Task Force. (1999) “RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1”. [s. L.]: IETF, 1999. 176 p. http://www.ietf.org/rfc/rfc2616.txt. Maio 2014.