Detector de Falhas Impact em uma Abordagem Requisição-Resposta Tolerante à Instabilidade do Sistema

Anubis Graciela de Moraes Rossetto[1], **Marcelo Felipe Guarani Fernandes**[1], **Cláudio F. R. Geyer**[2], **Luciana Arantes**[3], **Pierre Sens**[3]

[1]Instituto Federal Sul-rio-grandense (IFSUL), Campus Passo Fundo, Brasil

[2]Instituto de Informática – Universidade Federal do Rio Grande do Sul, Brasil

[3]Sorbonne Université, CNRS, Inria, LIP6, Paris, France

anubis.rossetto@passofundo.ifsul.edu.br
guaranimarcelo@gmail.com
geyer@inf.ufrgs.br
{luciana.arantes, pierre.sens}@lip6.fr

Abstract. *The Impact failure detector (FD), introduced in [Rossetto et al. 2015a] outputs a trust level value which expresses the degree of confidence in the system. An impact factor is assigned to each process and the trust level is equal to the sum of the impact factors of the processes not suspected of failure. Moreover, an input threshold parameter defines a lower bound value for the trust level, over which the confidence in the system is ensured. However, in the definition of the impact factor, every node has an impact factor value which does not change during execution. In this article, we propose a message-pattern implementation of the Impact FD (query-response) in which the concept of impact factor is extended to become dynamic. Its value is periodically re-evaluated based on the history of false suspicions of the nodes. Performance results of experiments conducted with real PlanetLab traces confirm that the dynamic impact factor increases significantly the confidence in the system, when compared to the original static impact factor.*

Resumo. *O detector de falhas Impact FD, introduzido em [Rossetto et al. 2015a], fornece como saída o grau de confiança no sistema (trust level). Um fator de impacto é atribuído a cada nodo e o* nível de confiança *é igual à soma dos fatores de impacto dos nodos não suspeitos de terem falhado. Ainda, um parâmetro de entrada* $threshold$ *define um valor limite do fator de impacto, sobre o qual o grau de confiança no sistema é assegurado. No entanto, na definição do Impact FD, cada nodo tem um valor de fator de impacto que não muda durante a execução. Neste artigo, apresentamos a proposta de implementação do Impact FD na abordagem baseada em padrão de mensagem (query-response) e na qual o conceito do fator de impacto é estendido para tornar-se dinâmico. Seu valor é reavaliado periodicamente com base no histórico de falsas suspeitas de falhas dos nodos. Os resultados da avaliação de desempenho com traces reais do PlanetLab confirmam que o fator de impacto dinâmico aumenta de forma significativa a confidência no sistema, quando comparado com o original fator de impacto estático.*

1. Introdução

Em sistemas distribuídos, falhas ocorrem e a detecção destas é crucial para a concepção de sistemas e aplicações tolerantes a falhas. Entretanto, em sistemas assíncronos não existe um limite máximo para a transmissão e execução de processos. Devido a esta ausência de limite, o problema do consenso não pode ser solucionado deterministicamente em sistemas assíncronos sujeitos a falhas por parada (crash) [Fischer et al. 1985]. Basicamente, esta impossibilidade resulta da dificuldade em determinar se um processo está falho ou se o processo e/ou a comunicação com este estão mais lentos.

A fim de contornar essa impossibilidade, Chandra e Toueg propuseram em [Chandra and Toueg 1996] a abstração do detectores de falhas (FD) não confiável. Um detector de falhas pode ser visto como um óraculo que fornece informações, nem sempre corretas, sobre falhas de processos, ou seja, pode suspeitar erroneamente de estarem falhos um ou mais processos no sistema. Ele é caracterizado por duas propriedades, *completude* (*completeness*) e *acurácia* (*acuracy*) [Chandra and Toueg 1996]. A completude está relacionada com a capacidade do detector suspeitar permanentemente de falhas dos processos, enquanto a acurácia diz respeito à capacidade de não suspeitar de processos corretos. No mesmo artigo, os autores classificam detectores de falhas de acordo com duas propriedades de completude e quatro propriedades de acurácia, as quais combinadas geram oito classes de detectores de falhas.

A maioria dos detectores é baseada no modelo binário no qual os processos monitorados são ou *"trusted"* ou *"suspected"*. Assim, muitos dos detectores de falhas existentes, como os definidos em [Chandra and Toueg 1996] e [Bertier et al. 2003], fornecem como saída o conjunto de processos que atualmente são suspeitos de terem falhado (*crashed*). Uma abordagem não binária é apresentada em [Hayashibara et al. 2004], o detector de falhas Accrual, que oferece como saída o nível de suspeita de cada processo em uma escala contínua.

Em [Rossetto et al. 2015a], [Rossetto et al. 2015b] e [Rossetto et al. 2016] propusemos o detector Impact (Impact FD) que fornece como saída um nível de confiança em relação a um conjunto S de processos monitorados. Esta pode ser considerada como o grau de confiança em S, ou seja, a confiança no conjunto de processos como um todo. Para este fim, um fator de impacto (*impact factor*) é atribuído a cada processo de S e um parâmetro limiar (*threshold*) define um valor limite, sobre o qual, o grau de confiança em S não é afetado. O fator de impacto indica a importância relativa do processo no conjunto S, enquanto o *threshold* oferece uma margem de flexibilidade para falhas e falsas suspeitas, permitindo assim uma maior tolerância a instabilidades no sistema. Consequentemente, apesar de falsas suspeitas ou mesmo falhas, mas devido ao valor do *threshold*, o sistema pode ainda atender ao grau de confiabilidade. Vale ressaltar que o *threshold* pode também caracterizar redundância de processos.

Uma implementação do Impact FD com uso de temporizadores e heartbeats para detectar falhas em processos foi apresentada em [Rossetto et al. 2015a] e [Rossetto et al. 2015b], assim como resultados de desempenho utilizando *traces* reais do PlanetLab [PlanetLab 2014]. Neste tipo de implementação, os processos enviam periodicamente mensagens de controle (*"heartbeat"*) indicando que estão operacionais. Se um processo não receber tais mensagens de um segundo processo, dentro de um

limite específico de tempo, aquele passa a suspeitar que este se encontra falho.

Já em [Rossetto et al. 2016], apresentamos uma implementação do Impact FD que adota a abordagem baseada em padrão de mensagem (*"query-response"*), proposta em [Mostéfaoui et al. 2003]. Se não falhar, um processo envia periodicamente uma requisição ("$query''$) aos outros processos e espera por um certo número α de respostas ("$responses''$). Os processos, cujas respostas não foram consideradas, são vistos como suspeitos de estarem falhos e a requisição corrente termina. As falsas suspeitas de processos são corrigidas quando da recepção, nas requisições seguintes, de respostas destes processos. Nesse artigo [Rossetto et al. 2016], introduzimos algumas propriedades que garantem que o sistema será sempre ou a termo confiável e as condições de estabilidade do sistema que asseguram a confiança no sistema. Porém, não são apresentados resultados de avaliação de desempenho mas apenas as provas de corretude dos algoritmos.

Nos artigos referentes ao Impact FD acima citados, consideramos que o valor do fator de impacto de cada processo, estaticamente atribuído na inicializaçao, não varia ao longo da execução. Entretanto, este valor pode ter sido superestimado ou subestimado. Além disso, durante os períodos de instabilidade da rede, o detector Impact FD pode suspeitar com maior frequência de processos cujo fatores de impacto são altos e cuja comunicação é mais lenta. Nestes casos, o Impact FD considerará, erroneamente, que o sistema é não confiável. A taxa destes enganos poderia ser evitada ou amenizada se os respectivos valores do fator de impacto dos processos fossem reavaliados durante o período de instabilidade. Uma outra situaçao é quando uma máquina ou mais se tornam temporariamente mais lentas devido à sobrecarga de processamento ("$overload''$). Assim, considerando que variações podem ocorrer na comunicação e no processamento dos sistemas atuais, neste artigo propomos uma implementaçao baseada na abordagem padrão de mensagem (*"query-response"*) em que o valor do fator de impacto atribuído a um processo é reavaliado periodicamente com base no histórico de falsas suspeitas. Experimentos de avaliação de desempenho foram realizados utilizando os mesmos *traces* do PlanetLab. Vale ressaltar que optamos pela requisição-respostas pois ainda não havíamos avaliado o desempenho do Impact FD com esta abordagem. Além disso, o término de uma requisição, cuja saída indica que o sistema não é confiável, apresenta-se como um ponto ideal para a reavaliação dos fatores de impacto dos processos de S.

O restante deste artigo está organizado nas seguintes seções: A Seção 2 motiva a necessidade do Impact FD e de um fator de impacto dinâmico. Na Seção 3, são descritos o modelo de sistema considerado e o Impact FD. Os algoritmos que implementam o Impact FD e permitem reavaliar dinamicamente o fator de impacto são apresentados na Seção 4, enquanto os resultados de avaliação preliminares conduzidos com *traces* reais do PlanetLab [PlanetLab 2014], na Seção 5. A Seção 6 discute alguns trabalhos relacionados. A Seção 7 apresenta a conclusão e trabalhos futuros.

2. Motivação

As características e flexibilidade do Impact FD pemitem que ele atenda a diferentes necessidades, podendo, desta forma, ser aplicado a diversos cenários distribuídos. Por outro lado, as versões anteriores do Impact FD não permitiam ao módulo detector

reavaliar, durante a execução da aplicação distribuída, o valor de impacto atribuído aos nodos.

Redes de sensores sem fio (RSSFs) ubíquas são frequentemente utilizadas para monitoração como, por exemplo, das condições físicas de regiões geográficas. Porém, sensores são sujeitos a falhas e restrições de energia. Neste caso, a redundância garante a cobertura da região e a conectividade da rede. Este tipo de cenário permite a definição de um *threshold* que é igual ao número mínimo de sensores necessários a manter a conectividade e a aplicação funcionando. No entanto, mesmo com a redundância, se sensores de um certo tipo apresentassem, por uma razão inesperada, problemas de hardware ou energético, eles começariam a serem suspeitos de falhas com maior frequência. Neste caso, o ideal seria reduzir o fator de impacto dos sensores deste tipo. Além disso, em algumas situações pode haver a necessidade de reconfigurar dinamicamente o grau de redundância. Para tanto, poderíamos também pensar em alterar o valor do fator de impacto dos sensores, cuja soma garante o mesmo valor do *threshold*.

Um outro exemplo seria um sistema com um servidor principal que oferece um determinado serviço com uma certa qualidade de serviço estimada (vazão, tempo de resposta, etc.) e servidores backup com qualidade de serviço muito menor que a oferecida pelo servidor principal. Em princípio, o servidor principal assegura a qualidade de serviço e, consequentemente, o *threshold* tem o mesmo valor que o valor do fator de impacto deste servidor. Entretanto se, por exemplo, devido à sobrecarga de processamento, a qualidade de serviço do servidor principal diminuir temporariamente, o ideal seria diminuir o seu fator de impacto e aumentar, se necessário, os dos servidores backups para que o sistema continue sendo confiável, oferecendo, desta forma, a qualidade de serviço esperada.

3. Detector de Falhas Impact

Consideramos um sistema distribuído que consiste em um conjunto finito de processos $\Pi = \{q_1, \ldots, q_n\}$, com $|\Pi| = n, (n \geq 2)$. As falhas são apenas por parada (*crash*).

Assumimos a existência de algum tempo global denotado T. Um padrão de falha é uma função $F : T \rightarrow 2^{\Pi}$, onde $F(t)$ é o conjunto de processos que falharam antes ou no tempo t. A função $correct(F)$ denota o conjunto de processos corretos, ou seja, aqueles que nunca pertencem ao padrão de falha (F), enquanto que $faulty(F)$ denota o conjunto de processos falhos, ou seja, o complemento de $correct(F)$ em relação a Π. Um processo $p \in \Pi$ monitora um conjunto S de processos de Π. Denotamos $correct(F_S) = correct(F) \cap S$ and $faulty(F_S) = faulty(F) \cap S$.

Os canais de comunicação são assíncronos, bidirecionais e confiáveis, isto é, não há perda de mensagens e estas não são nunca alteradas nem duplicadas, existindo um canal de q ($\forall q \in S$) para p.

O **Impact FD** pode ser definido como um detector de falhas não confiável que fornece uma saída relacionada ao nível de confiança com relação a um conjunto de processos. Se o nível de confiança (*trust level*), fornecido pelo detector, é igual ou maior do que um determinado limiar (*threshold*), definido pelo usuário, a confiança no conjunto de processos é assegurada. Podemos dizer, neste caso, que o sistema é confiável.

Denotamos $FD({I_p}^S)$ o módulo Impact FD do processo p e S é um conjunto de processos de Π.

Fator de Impacto (impact factor): Cada processo $q \in S$ tem um fator de impacto (*impact factor*) associado ($I_q | I_q > 0 : I_q \in \mathbb{R}$). Definimos então S^* como o conjunto composto de tuplas $\langle id, I \rangle$, sendo id o identificador do processo e I o valor do fator de impacto do processo.

Nível de confiança (trust level): Quando invocado em p, o Impact FD (${I_p}^S$) retorna o valor ${trust\_level_p}^S$. O ${trust\_level_p}^S$ é um valor que expressa o nível de confiança (*trust level*) que p tem no conjunto S. Denotamos ${trusted_p}^S(t)$ o conjunto de processos de S que são considerados não falhos por p no tempo $t \in T$. Definimos então ${trust\_level_p}^{S^*}(t) = Trust\_level({trusted_p}^S, S^*)$, onde a função $Trust\_level(set)$ retorna a soma do fator de impacto dos elementos $\langle id_q, I_q \rangle$ de S^*, tal que $id_q \in trusted_p^S$.

Margem de falhas (threshold): Um limiar aceitável de falhas, denotado ${threshold}^S > 0$, caracteriza o grau aceitável de flexibilidade de falha em relação ao conjunto S^*. O ${threshold}^S$ está relacionado com o nível mínimo de confiança requerido e é usado pela aplicação para verificar a confiança nos processos de S. Se ${trust\_level}^{S^*}(t) \geq {threshold}^S$, S é considerado confiável (*trusted*) no tempo t por p, ou seja, a confiança de p em S não foi comprometida; caso contrário, S é considerado não confiável (*untrusted*).

Dois pontos merecem ser ressaltados: (1) O $impact\ factor$ e o ${threshold}^S$ tornam a estimativa da confiança em S flexível. Por exemplo, pode acontecer que alguns processos em S a $t \in T$ estejam falhos ou suspeitos de falhos, mas S ainda é considerado confiável por p a t; (2) o ${threshold}^S$ aumenta a tolerância de S a falsas suspeitas. Também é interessante notar que o Impact FD é facilmente configurável de acordo com as necessidades do ambiente. O ${threshold}^S$ pode ser ajustado de forma a fornecer um monitoramento com maior ou menor rigor.

Exemplo: A Tabela 1 apresenta um exemplo, considerando um conjunto S^* composto de quatro processos, sendo o fator de impacto de todos os processos igual a 1 ($S^* = \{\langle q_1, 1\rangle, \langle q_2, 1\rangle, \langle q_3, 1\rangle, \langle q_4, 1\rangle\}$). O valor para o ${threshold}^S = 2$ define que o conjunto S deve ter pelo menos dois processo corretos. A tabela mostra diferentes saídas para o Impact FD (${I_p}^S$) dependendo das falhas dos processos: o conjunto S é considerado confiável se $trust\_level(t) \geq {threshold}^S$.

Tabela 1. Exemplo de saída do Impact FD (${I_p}^S$): S^* possui quatro processos

t	F(t)	${trusted_p}^S(t)$	${trust\_level_p}^{S^*}(t)$	Status at t
1	$\{q_2\}$	$\{q_1, q_3, q_4\}$	{3}	Trusted
2	$\{q_2, q_4\}$	$\{q_1, q_3\}$	{2}	Trusted
3	$\{q_1, q_2, q_4\}$	$\{q_3\}$	{1}	Untrusted

$S^* = \{\langle q_1, 1\rangle, \langle q_2, 1\rangle, \langle q_3, 1\rangle, \langle q_4, 1\rangle\}$
${threshold}^S = 2$

Propriedade de flexibilidade: denota a possibilidade do detector fornecer diferentes respostas que levam a um estado confiável sobre o conjunto de processos. Considere PS como o conjunto que contém todos os possíveis subconjuntos de elementos que

satisfazem o *threshold* definido:

$$PS = PowerSet(S^*, threshold)$$

A função *PowerSet* gera o conjunto das partes[1] para cada tupla de S^*. Após, são filtrados apenas os subconjuntos de S^* cuja a soma dos seus elementos é maior ou igual ao seu *threshold*. Vejamos um exemplo:

$S^* = \{\langle q_1, 2\rangle, \langle q_2, 4\rangle, \langle q_3, 3\rangle\}$; $threshold^S = 3$
$PS = Powerset(S^*, threshold^S) = \{\langle q_2, 4\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_1, 2\rangle, \langle q_2, 4\rangle\}, \{\langle q_1, 2\rangle, \langle q_3, 3\rangle\}, \{\langle q_2, 4\rangle \langle q_3, 3\rangle\}$, $\{\langle q_1, 2\rangle, \langle q_2, 4\rangle, \langle q_3, 3\rangle\}$

Em [Rossetto et al. 2015a], [Rossetto et al. 2015b] e [Rossetto et al. 2016], apresentamos outras propriedades do Impact FD, como $(Eventual)\ Impact\ Threshold$ e $(Eventual)\ PS-acessible$ que possibilitam asegurar as propriedades de *completude* e *acurácia* do Impact FD, definir diferentes classes, assim como a possibilidade de configurar subconjuntos do conjunto S^* com diferentes valores de *threshold*. Por não serem utilizadas neste artigo, todas estas funcionalidades do Impact FD não serão aqui descritas. Para maiores detalhes, vide os artigos mencionados.

4. Algoritmo Impact FD Requisição-Resposta

Em [Mostéfaoui et al. 2003], os autores propõem uma implementação para detectores de falhas que faz uso de um mecanismo de requisição-resposta (*query-response*). Nesta abordagem, um processo p transmite uma mensagem $QUERY$ para os n nodos que ele monitora e então espera as respostas ($RESPONSE$) correspondentes de α processos ($\alpha \leq n$, tradicionalmente, $\alpha = n - f$, onde f é o número máximo de falhas). As outras respostas associadas à consulta, se houver, podem ser eliminadas. A consulta termina quando p receber α respostas. Os demais processos, cujas respostas não foram consideradas, são vistos como suspeitos de estarem falhos. Um processo p, se não falhar, transmite mensagens $QUERY$ repetidamente. Se, numa rodada, p recebe uma resposta de um processo suspeito anteriormente, então p remove este processo de sua lista de suspeitos. Note que nesta abordagem, o conjunto de processos que responderam a mensagem $QUERY$ não é fixo, isto é, pode mudar a cada rodada.

Os Algoritmos 1 e 2 apresentam a implementação na abordagem requisição-resposta do Impact FD do processo monitor ($p \in correct(F)$ e $p \notin S$) com relação ao conjunto S. O processo p repetidamente emite requisições usando a primitiva *broadcast(m)*, que envia uma mensagem $QUERY$ para cada q, $\forall q \in S$. O intervalo de tempo entre duas rodadas consecutivas de mensagens $QUERY$ é finito e arbitrário.

A recepção da mensagem $QUERY$ é tratada pela tarefa Task T1 (Algoritmo 1), que é executada por cada processo $q \notin faulty(F_S)$. Após a recepção da mensagem $(QUERY, r_p)$, onde r_p é o identificador da rodada (linha 2), q responde a p com uma mensagem $RESP$ identificada pelo mesmo valor de rodada r_p (linha 3).

No algoritmo 2, p recebe como valores de entrada S^*, o valor $threshold^S$ e o número máximo de mensagens (α) a esperar do conjunto S. O algoritmo possui três tarefas. Na inicialização, os processos de S são atribuídos a $trusted$ e a função

[1] O conjunto das partes de qualquer S é o conjunto de todos os subconjuntos de S.

Algoritmo 1 Algoritmo Impact FD para $q \in S$

```
1: Begin
   Task T1
2:     Upon reception of (QUERY, r_p) from p do
3:         send(RESP, r_p) to p
4:     end
5: End
```

$PowerSet$ é executada para gerar o conjunto PS que contém todos os subconjuntos possíveis formados por processos de S que satisfaçam o $threshold^S$. A cada rodada, a variável $tmp\_trusted$ contém o identificador e o fator de impacto dos processos que responderam à consulta atual.

Na tarefa TASK T1 de p há um laço infinito. Inicialmente p envia a mensagem *(QUERY,* r_p*)* a todos os processos de S (linha 11). Então, a cada rodada (r_p), p espera por pelo menos α respostas ou até $tmp\_trusted$ ser um subconjunto de PS (i.e., contém os processos cuja soma do valor do fator de impacto satisfaz o $threshold^S$, linha 12). O fator de $tmp\_trusted$ não pertencer ao *PowerSet* significa que o nível de confiança não foi satisfeito nesta rodada e que os valores de impacto dos processos necessitam, talvez, serem reavaliados. Para tanto, a função $update\_impact(S^*)$ é executada (linha 14). Note que a implementação desta função depende do critério utilizado pelo detector para atualizar o fator de impacto dos processos. Na Seção 5, este é reavaliado com base no histórico de falsas suspeitas. A função $update\_impact(S^*)$ deve também gerar novamente o conjunto PS. Finalmente, o conjunto $tmp\_trusted$ é salvo em $trusted$ e o contador da rodada (r_p) é incrementado (linha 18).

A tarefa TASK T2 é responsável pela recepção das mensagens *(RESP,* r_q*)* enviadas por um processo $q \notin faulty(F_S)$. Se o número da rodada r_q da mensagem *RESP* é igual a r_p, então q é adicionado ao conjunto $tmp\_trusted$.

A função *Impact( )* (linha 25) da tarefa TASK T3 computa e retorna o nível de confiança relacionado aos processos considerados não falhos (linha 26).

5. Avaliação de Desempenho

Nessa seção, inicialmente, descrevemos o ambiente no qual os experimentos foram conduzidos. Apresentamos então os resultados obtidos nos diferentes experimentos bem como as suas respectivas discussões.

O objetivo é avaliar a QoS do Impact FD, ou seja, a capacidade do Impact FD em evitar falsas suspeitas e adaptar o fator de impacto, considerando uma implementação com abordagem requisição-resposta. Para tanto, o Algoritmo 2, apresentado na seção 4, foi implementado, assim como a função $update\_impact(S^*)$ cujo critério de avaliação do fator de impacto dos processos é baseado no histórico de falsas suspeitas.

Da mesma forma que os experimentos realizados em [Rossetto et al. 2015a] e [Rossetto et al. 2015c], os testes conduzidos neste trabalho foram baseados em arquivos de traços reais coletados de dez sites do PlanetLab [PlanetLab 2014], durante uma semana. Cada site envia mensagens de heartbeat para os outros sites a uma taxa de um *heartbeat* a cada 100 ms (intervalo de envio). No entanto, uma vez que a implementação utilizada neste trabalho considera a abordagem requisição-resposta, foi simulado

Algoritmo 2 Implementação requisição-resposta para p

1: **Begin**
Input
2: S^*, $threshold^S$, α
Init
3: $r_p = 0$
4: $trusted = \emptyset$
5: $tmp\_trusted = \emptyset$
6: **for** $i = 1$ *to* $|S^*|$ **do**
7: $trusted = trusted \cup \{q_i\}$
8: **end for**
9: $PS = PowerSet(S^*, threshold^S)$

Task T1
10: **loop**
11: $broadcast(QUERY, r_p)$
12: $wait\ until\ (|tmp\_trusted| \geq \alpha)\ or\ (tmp\_trusted \in PS)$
13: **if** $tmp\_trusted \notin PS$ **then** ▷ Se não satisfaz o threshold
14: $update\_impact(S^*)$
15: **end if**
16: $trusted = tmp\_trusted$
17: $tmp\_trusted = \emptyset$
18: $r_p = r_p + 1$
19: **end loop**

Task T2
20: **Upon reception of** $(RESP, r_q)$ from q **do**
21: **if** $r_q = r_p$ **then**
22: $tmp\_trusted \cup \{q\}$
23: **end if**
24: **end**

Task T3
25: **Upon invocation of** $Impact()$ **do**
26: **return** $Trust\_level(trusted, S^*)$
27: **end**

28: **End**

o envio da mensagem $QUERY$ pelo site monitor e as mensagens de heartbeat são consideradas como sendo a mensagem $send(RESP, r_p)$ dos processos q para p (Algoritmo 1). Optou-se por usar esses arquivos de traços do PlanetLab tendo em vista que contêm uma grande quantidade de dados relativos ao envio e recepção de mensagens, incluindo períodos instáveis de links. Ressalta-se que, como os tempos de envio e chegada de cada mensagem são registrados nos arquivos de traços, todos os experimentos são realizados com exatamente os mesmos dados. Destaca-se ainda que o $site\,2$ parou de enviar mensagens definitivamente depois de aproximadamente 48 horas.

Os testes de avaliação comparam os resultados obtidos na execução do Algoritmo 2 da seção 4 com fator de impacto fixo e dinâmico. Para a versão com valor de impacto fixo, a função $update\_impact(S^*)$ nunca é chamada.

Considerando os 10 sites dos traços, definiu-se o conjunto *S = {1, 2, 3, 4, 5, 6, 7,*

8, 9 }. O site *0* é o monitor (p) e não pertence a S. O valor do fator de impacto atribuído a cada site é igual a 10, ou seja, a soma do fator de impacto dos sites é igual a 90. No caso do impacto dinâmico, esse valor é reavaliado, como apresentado a seguir.

Para cada processo de S, é mantido um histórico com a quantidade de mensagens $RESP$ recebidas em resposta às mensagens $QUERY$ enviadas pelo site monitor. O histórico recente de respostas recebidas é armazenado numa janela, reiniciada a cada 10 falsas suspeitas.

5.1. Fator de Impacto Dinâmico

Nos experimentos referentes ao fator de impacto dinâmico, a cada ocorrência de falsa suspeita ($trust\_level^S < threshold^{S^*}$), o fator de impacto de cada processo de S é reavaliado, considerando o seu histórico de mensagens recebidas. Na rotina de atualização $update\_impact(S^*)$, é atribuído um fator de impacto mais alto aos processos com maior número de mensagens recebidas e um fator de impacto mais baixo aos processos com menor número de mensagens. Dessa forma, os sites que tem maior propensão a não responderem a mensagem de QUERY, passam a ter menor impacto no nível de confiança do conjunto ($trust\_level$), tendendo a apresentar, consequentemente, um menor número de falsas suspeitas do conjunto de processos de S.

Para avaliar o fator de impacto dinâmico, adotamos a seguinte estratégia:

- Os processos de S são divididos em 3 grupos: S_A, S_B e S_C. O grupo S_A contém sites com o maior número de mensagens no histórico. No grupo S_C estão os sites com menor número de mensagens e no grupo S_B ficam os demais sites. O número de sites de cada grupo e o seu fator de impacto, são definidos a seguir.
- $sites_A$ - é o número de processos que serão atribuídos ao grupo S_A, ou seja, 30% dos processos de S (com truncamento): $sites_A = \lfloor |S| * 30\% \rfloor$;
- $sites_B$ - é número de processos atribuídos ao grupo S_B: $sites_B = |S| - (|S_A| + |S_C|)$;
- $sites_C$ - é o número de processos que serão atribuídos ao grupo S_C: $sites_C = \lfloor |S| * 30\% \rfloor$;
- I - é o fator de impacto atribuído inicialmente a todos os processos;
- i_A - é o fator de impacto atribuído aos sites do grupo S_A: $i_A = I * 1.5$;
- i_B - é o fator de impacto atribuído aos sites do grupo S_B: $i_B = I$;
- i_C - é o fator de impacto atribuído aos sites do grupo S_C: $i_C = \frac{I}{2}$.

Tomando por base o conjunto S definido para os experimentos, os valores ficam assim definidos: $sites_A = 3; sites_B = 3; sites_C = 3; I = 10; i_A = 15; i_B = 10; i_C = 5;$.

Experimento 1 - Número de falsas suspeitas - α = 5 e 7

Este experimento avalia o número de falsas suspeitas considerando diferentes valores de $threshold$ (65, 70, 75, 80 e 85) para os valores de α = 5 e 7.

A Figura 1 mostra os resultados dos testes obtidos. Com relação ao número de falsas suspeitas, pode-se observar que, com o fator de impacto dinâmico, o número de enganos cometidos pelo Impact FD foi consideravalmente menor na maioria das configurações para os dois valores de α. Apenas em duas configurações não se obteve redução no número de falsas suspeitas: com α = 5 e $threshold$ = 80 e 85. Nestas configurações, o número de enganos foi o mesmo. Isso decorre do fato que α está definido

com um valor baixo, fazendo com que o critério $|tmp\_trusted| \geq \alpha$ seja satisfeito em detrimento da condição que trata do $threshold$ ($tmp\_trusted \in PS$), especialmente com um alto valor de $threshold$, como é o caso. Assim, quando chegam mensagens de 5 sites, a rodada é finalizada, independentemente da soma do fator de impacto dos processos satisfazer o $threshold$.

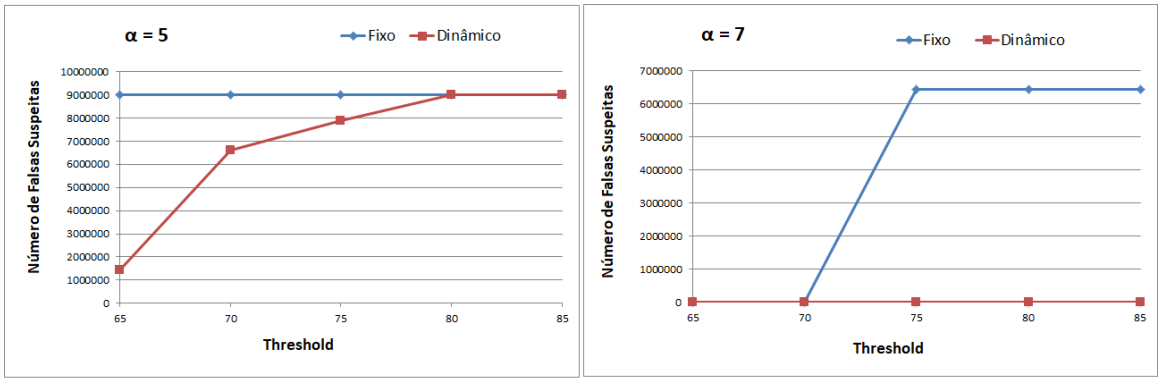

Figura 1. Gráficos Comparativos do Fator de Impacto Fixo X Dinâmico - Número de falsas suspeitas (α = 5 e 7)

Experimento 2 - Número de falsas suspeitas - $threshold$ = 70 e 85

A Figura 2 apresenta os gráficos correspondentes ao número de falsas suspeitas para os testes, considerando $threshold$ = 70 e 85, ou seja, um valor mais flexível e outro mais restritivo. Nesse caso, α variou de 5 até 8. Com base nesses resultados, pode-se destacar que o fator de impacto dinâmico tem uma influência significativa na redução do número de falsas suspeitas, principalmente em cenários onde a restrição de erros é menor, ou seja, se tem valores de α e $threshold$ mais agressivos, o que induz a propensão a enganos pelo detector de falhas. De outro lado, também ressalta-se que nos resultados obtidos, o fator de impacto dinâmico não implicou em diferença significativa no tempo de detecção (T_D). Consideramos que o tempo de detecção (T_D) é o tempo desde o envio da mensagem $QUERY$ pelo site monitor até o fim da rodada.

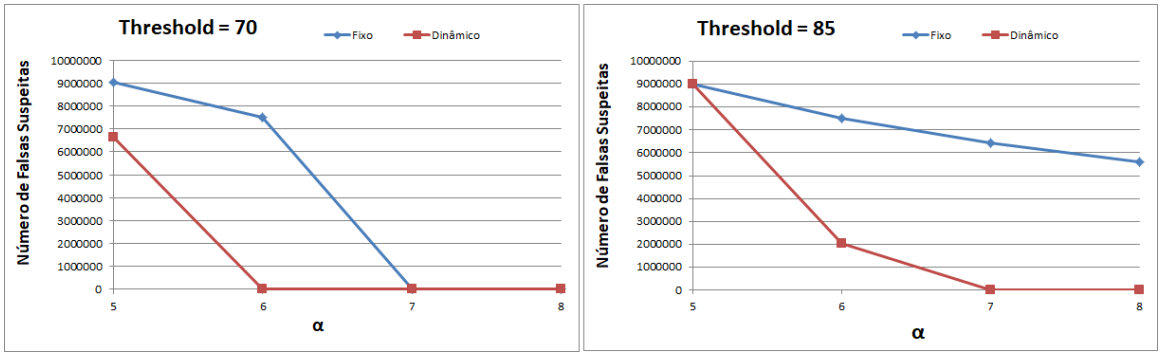

Figura 2. Gráficos Comparativos do Fator de Impacto Fixo X Dinâmico - Número de falsas suspeitas ($threshold$ = 70 e 85)

Experimento 3 - % Falsas Suspeitas

A Figura 3 mostra os gráficos correspondentes ao percentual de falsas suspeitas para os valores de $threshold$ = 70 e 85 e variando α (5, 6, 7 e 8). Com o fator de impacto fixo é possível observar que quando α é definido com um valor baixo, o conjunto S sempre será não confiável. Por exemplo, para $threshold$ = 70, com os valores de α = 5 e 6, em todas as rodadas $tmp\_trusted \notin PS$, ou seja, o $threshold$ não foi satisfeito. Já com o fator de impacto dinâmico, mesmo com α = 5, o Impact FD reduziu o percentual de falsas suspeitas, isto significa que com respostas de apenas 5 sites, em alguns casos, o sistema é confiável. Com um valor mais restritivo para o $threshold$ (85), o percentual de falsas suspeitas para todos os valores de α foi de 100% com o fator de impacto fixo, enquanto que com o fator de impacto dinâmico se obteve resultados melhores conforme o valor de α aumenta.

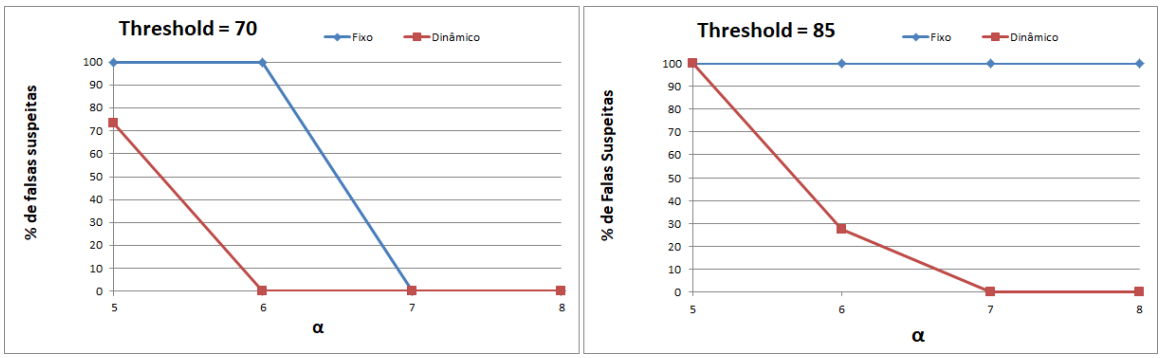

Figura 3. Gráficos Comparativos do Fator de Impacto Fixo X Dinâmico - % de falsas suspeitas ($threshold$ = 70 e 85)

6. Trabalhos Relacionados

Várias implementações de detectores da classe $\diamond S$ ou para eleição de líder a termo (Ω) na literatura, como [Mostéfaoui et al. 2003], [Arantes et al. 2013], [Greve et al. 2012], são baseadas na estratégia de *query-response* e incluem hipóteses adicionais de sincronismo para a estabilidade da rede a termo.

A maioria dos detectores de falhas não confiável da literatura é baseada no modelo binário e fornece como saída um conjunto de identificadores de processos, que, geralmente, expressa o conjunto de processos atualmente suspeitos de terem falhado ([Chandra and Toueg 1996], [Bertier et al. 2003]). O detector de falhas Accrual ϕ [Hayashibara et al. 2004] propõe uma abordagem em que a saída do detector é, para cada processo, o nível de suspeita em uma escala contínua, em vez de fornecer informações de natureza binária (*trusted* ou *suspected*). Assim, ele tem por objetivo dissociar o monitoramento da interpretação. O nível de suspeita do processo q, monitorado por p, expressa o grau de confiança de p na asserção de que q falhou. Se este realmente falhou, o valor do nível de suspeita acumula ao longo do tempo e, consequentemente, tende ao infinito. Em [Satzger et al. 2007], os autores apresentam uma nova abordagem de Accrual FD com base em uma estimativa acerca da densidade do histograma de tempo de chegada amostrado. Levando em consideração uma amostra dos tempos entre chegadas, bem como o tempo do último *heartbeat* recebido, o algoritmo calcula a probabilidade de que não haja mais mensagens de *heartbeat* para chegar, ou seja, o processo falhou.

[Leners et al. 2013] propõe um serviço para reportar falhas às aplicações, encapsulando a incerteza. Ele parte da premissa de que os aplicativos deveriam ter informações sobre as falhas para tomar ações de recuperação adequadas. O objetivo é fornecer relatórios de estado vinculados à detecção de falhas com uma abstração que relata o grau de certeza. O trabalho fornecido em [Brun et al. 2011] apresenta uma nova técnica de redundância, baseada em votação, a fim de melhorar a confiabilidade em sistemas distribuídos, considerando que cada nodo tem uma probabilidade de ser byzantino. Considerando tais valores, os autores calculam o número mínimo de máquinas necessário para que o sistema como um todo ofereça confiabilidade igual ou maior a um limiar (*threshold*).

Em [Véron et al. 2015], os autores implementaram um detector de falhas para sistemas de larga escala utilizando um mecanismo de reputação que permite aos nodos trocarem visões parciais do sistema, explorando, assim, as informações sobre o comportamento (reputação) dos nodos com o objetivo de obter uma melhor qualidade da detecção de falhas.

Muitos trabalhos na literatura propõem diferentes estratégias para estimar a instabilidade da rede e a chegada dos heartbeats, calibrando, consequentemente, o período entre envios de hearbeats e o valor dos timeouts. Bertier et al. [Bertier et al. 2003] introduziram um detector para redes locais cuja estimação da chegada do heartbeat inclui o algoritmo de Chen e uma estimativa dinâmica baseada no algoritmo de Jacobson, utilizada para estimar o tempo para retransmissão de uma mensagem no protocolo TCP. Explorando a teoria de controle e qualidade de serviço (QoS), o detector de falhas autonômico AFD [de Sá and Macêdo 2010] configura dinamicamente o período de monitoramento e valor de timeout. O detector de falhas 2W-FD [Tomsic et al. 2015] foi concebido para tolerar redes instáveis como variação de latência e contenção em roteadores ou nodos. Para tanto armazena os heartbeats em janelas.

7. Considerações Finais

Na definição do Impact FD apresentada em [Rossetto et al. 2016], cada nodo possui um valor de fator de impacto estático enquanto que neste trabalho, foi apresentada a proposta de um algoritmo do Impact FD baseado na abordagem requisição-resposta que emprega o fator de impacto dinâmico, ou seja, o valor do fator de impacto de um nodo pode variar durante a execução, dependendo do histórico de estabilidade do nodo. Os resultados da avaliação de desempenho mostram que o fator de impacto dinâmico diminui significativamente o número de falsas suspeitas do Impact FD quando comparado com o fator de impacto fixo. Estes resultados, mesmo que preliminares, são bastante promissores. Eles confirmam o grau de aplicabilidade flexível do Impact FD, sendo tanto as falhas quanto as falsas suspeitas mais toleradas com o fator de impacto dinâmico em cenários nos quais a aplicação estiver interessada no grau de confiança no sistema como um todo.

Como trabalho futuro, a curto prazo, pretendemos realizar experimentos mais exaustivos de avaliação de desempenho, inclusive avaliando o impacto que a falha de um nodo com um valor de fator de impacto aumentado pode ter na confiança do sistema em relação a um valor de fator de impacto que não varia. A médio prazo, queremos considerar outros critérios/parâmetros para reavaliar o fator de impacto dinâ-

mico como, por exemplo, energia da bateria do nodo (no caso de rede de sensores, MANET ou IOT), a carga atual de processamento do nodo, a probablidade de falhas, etc., que variam ao longo da execução e podem ter um impacto na confiança no sistema. Para tanto, seria necessário oferecer uma interface para que a aplicação e/ou sistema operacional do nodo pudesssem informar o módulo detector sobre o novo valor do critério/parâmetro em questão além de incluí-lo nas mensagens de *response* para o nodo monitor. Além disso, poderíamos também oferecer a possibilidade de dinamicamente reconfigurar o valor do $threshold^S$ segundo algum critério/parâmetro. Por exemplo, dependendo do período do dia, o monitoramento pode necessitar um maior rigor e, neste caso, o valor do $threshold^S$ teria que ser aumentado.

Outra direção de estudo é generalizar o cálculo do nível de confiança ($trust\_level$), bem como a sua comparação com o $threshold$. Com isso, a função $Trust\_level\,(trusted, S^*)$ executaria uma operação sobre o fator de impacto dos processos confiáveis que não a soma (multiplicação, média, etc.) e o $threshold^S$ não seria necessarimente o limite inferior (limite superior, igualdade, etc.). Por exemplo, se o fator de impacto corresponde à probabilidade de comportamento malicioso, o $trust\_level$, neste caso, expressaria a probabilidade de que todos os nodos do sistema se comportem maliciosamente. Assim, a operação de soma seria substituída pela operação de multiplicação.

Por fim, para poder utilizar o Impact FD em sistemas em larga escala e com nodos móveis (IOT, Manet, etc.), teríamos que modificar a forma de progagar as mensagens pela rede até o nodo monitor pois, nestes casos, a rede não seria totalmente conectada e nem sempre conexa.

Referências

[Arantes et al. 2013] Arantes, L., Greve, F., Sens, P., and Simon, V. (2013). Eventual leader election in evolving mobile networks. In *17th International Conference Principles of Distributed Systems, OPODIS*, pages 23–37, Nice, France. Springer.

[Bertier et al. 2003] Bertier, M., Marin, O., and Sens, P. (2003). Performance analysis of a hierarchical failure detector. In *2003 International Conference on Dependable Systems and Networks (DSN)*, pages 635–644, San Francisco, CA, USA. IEEE Computer Society.

[Brun et al. 2011] Brun, Y., Edwards, G., Bang, J. Y., and Medvidovic, N. (2011). Smart redundancy for distributed computation. In *International Conference on Distributed Computing Systems, ICDCS*, pages 665–676, Minneapolis, Minnesota, USA. IEEE Computer Society.

[Chandra and Toueg 1996] Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267.

[de Sá and Macêdo 2010] de Sá, A. S. and Macêdo, R. J. A. (2010). Qos self-configuring failure detectors for distributed systems. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 126–140, Amsterdam, The Netherlands. Springer Berlin Heidelberg.

[Fischer et al. 1985] Fischer, M., Lynch, N., and Paterson, M. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382.

[Greve et al. 2012] Greve, F., Sens, P., Arantes, L., and Simon, V. (2012). Eventually strong failure detector with unknown membership. *Comput. J.*, 55(12):1507–1524.

[Hayashibara et al. 2004] Hayashibara, N., Defago, X., Yared, R., and Katayama, T. (2004). The φ accrual failure detector. In *23rd International Symposium on Reliable Distributed Systems SRDS*, pages 66–78, Florianopolis, Brazil. IEEE Computer Society.

[Leners et al. 2013] Leners, J. B., Gupta, T., Aguilera, M. K., and Walfish, M. (2013). Improving availability in distributed systems with failure informers. In *10th USENIX Symposium on Networked Systems Design and Implementation, NSDI*, pages 427–441, Lombard, IL, USA. USENIX Association.

[Mostéfaoui et al. 2003] Mostéfaoui, A., Mourgaya, E., and Raynal, M. (2003). Asynchronous implementation of failure detectors. In *International Conference on Dependable Systems and Networks (DSN)*, pages 351–360, San Francisco, CA, USA. IEEE Computer Society.

[PlanetLab 2014] PlanetLab (2014). Planetlab. http://www.planet-lab.org. "Online. Access date: September 16, 2016".

[Rossetto et al. 2015a] Rossetto, A., Geyer, C., Arantes, L., and Sens, P. (2015a). A failure detector that gives information on the degree of confidence in the system. In *Symposium on Computers and Communication*, pages 532–537, Larnaca, Cyprus. IEEE Computer Society.

[Rossetto et al. 2015b] Rossetto, A. G., Arantes, L., Sens, P., and Geyer, C. F. R. (2015b). Impact: Um detector de falhas baseado na relevância dos processos e no grau de confiança no sistema. In *SBRC*.

[Rossetto et al. 2015c] Rossetto, A. G., Geyer, C. F., Arantes, L., and Sens, P. (2015c). Impact: an unreliable failure detector based on processes' relevance and the confidence degree in the system. Technical report. INRIA Nº hal-01136595.

[Rossetto et al. 2016] Rossetto, A. G., Geyer, C. F. R., Arantes, L., and Sens, P. (2016). Implementing a flexible failure detector that expresses the confidence in the system. In *LADC*, pages 61–70.

[Satzger et al. 2007] Satzger, B., Pietzowski, A., Trumler, W., and Ungerer, T. (2007). A new adaptive accrual failure detector for dependable distributed systems. In *ACM Symposium on Applied Computing (SAC)*, pages 551–555, Seoul, Korea. ACM.

[Tomsic et al. 2015] Tomsic, A., Sens, P., Garcia, J., Arantes, L., and Sopena, J. (2015). 2w-fd: A failure detector algorithm with qos. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 885–893, Hyderabad, India. IEEE.

[Véron et al. 2015] Véron, M., Marin, O., Monnet, S., and Sens, P. (2015). Repfd-using reputation systems to detect failures in large dynamic networks. In *44th International Conference on Parallel Processing, ICPP*, pages 91–100, Beijing, China. IEEE Computer Society.