Impact: Um detector de falhas baseado na relevância dos processos e no grau de confiança no sistema

Anubis Graciela de Moraes Rossetto[1], Luciana Arantes[2], Pierre Sens[2], Cláudio F. R. Geyer[1]

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil,

[2]Sorbonne Universités, UPMC Université. Paris 06, CNRS, Inria, LIP6

{agmrossetto, geyer}@inf.ufrgs.br

{luciana.arantes, pierre.sens}@lip6.fr

Abstract. *This work proposes a new and flexible unreliable failure detector, denoted the Impact Failure Detector (FD), whose output is related to the trust level of a set of processes. By expressing the relevance of each node by an impact factor value as well as a margin of acceptable failures of the system, the Impact FD enables the user to tune the failure detection configuration in accordance with the requirements of the application: in some scenarios, the failure of low impact or redundant nodes does not jeopardize the confidence in the system, while the crash of a high impact factor one may seriously affect it. A softer or stricter monitoring is thus possible. Performance evaluation results using real PlanetLab [PlanetLab 2014] traces confirm the degree of flexibility of our failure detector and, due to the margin of failure, the number of false responses may be reduced when compared to traditional unreliable failure detectors.*

Resumo. *Este trabalho propõe um novo detector de falhas não confiável chamado Impact Failure Detector (FD), cuja saída está relacionada ao nível de confiança em um conjunto de processos. Ao expressar a relevância de cada processo por um valor de fator de impacto, bem como por uma margem de falhas aceitáveis do sistema, o Impact FD permite ao usuário ajustar a configuração do detector de falhas de acordo com os requisitos da aplicação: em certos cenários, a falha de um processo de baixo impacto ou redundante não compromete a confiança no sistema, enquanto a falha de um processo de alto fator de impacto pode afetá-la seriamente. Assim, é possível um monitoramento com maior ou menor rigor. Os resultados da avaliação de desempenho usando traces reais do PlanetLab [PlanetLab 2014] confirmam o grau de flexibilidade do nosso detector de falhas e, devido à margem de falhas, o número de falsas respostas pode ser reduzido quando comparado a detectores de falhas tradicionais.*

1. Introdução

Um detector de falhas não confiável pode ser visto como um óraculo que fornece informações, nem sempre corretas, sobre falhas de processos. 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 está presente 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.

Neste artigo apresentamos um novo detector de falhas não confiável, o Impact Failure Detector (Impact FD) que fornece como saída um nível de confiança em relação a um conjunto S de processos monitorados. A saída 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 à instabilidade do sistema. Por exemplo, em uma rede instável pode haver muitas falsas suspeitas ou mesmo falhas, mas, devido ao valor do *threshold*, o sistema atende ao grau de confiabilidade. No caso do Impact FD indicar que a confiabilidade em S está abaixo do limite definido pelo *threshold*, o usuário pode decidir qual medida tomar (urgente ou não, em relação ao nível de confiança indicado pelo Impact FD). Além disso, no conceito do Impact FD, os processos monitorados podem ser agrupados com base em algum critério como, por exemplo, tipo dos processos (nodos, sensores) ou relevância destes. Observe que, juntamente com o *threshold*, tal abordagem de grupo pode caracterizar redundância de processos.

As características e flexibilidade do Impact FD permitem que ele atenda a diferentes necessidades, podendo, desta forma, ser aplicado a diversos cenários distribuídos. Por exemplo, em redes de sensores sem fio (RSSFs) ubíquas, frequentemente utilizadas para monitorar as condições físicas de regiões geográficas, onde os sensores são de diferentes tipos (de controle de humidade, de temperatura, etc.) e o número de sensores distribuídos na região varia segundo estes tipos. Sensores são sujeitos a falhas e a redundância, neste caso, garante a cobertura da região e a conectividade da rede. Assim, poderíamos considerar a região como um único conjunto cujos sensores do mesmo tipo são agrupados em subconjuntos. Esta abordagem permite a definição de um *threshold* que é igual ao número mínimo de sensores que cada subconjunto deve ter, a fim de manter a conectividade e a aplicação funcionando. Além disso, em algumas situações pode haver a necessidade de reconfigurar dinamicamente o grau de redundância. Neste caso, basta mudar o valor do *threshold*. Um outro exemplo seria um sistema com um servidor principal que oferece um determinado serviço com uma certa qualidade de serviço X (vazão, tempo de resposta, etc.). Se este falhar, N servidores backup tem capacidade para substituí-lo, visto que cada backup oferece o mesmo serviço mas com uma qualidade de serviço X/N. Neste cenário, tanto o fator de impacto do servidor principal quanto o *threshold* teriam o valor de $N * Iback$, sendo $Iback$, o valor de impacto dos servidores backups, ou seja, o sistema não é confiável no caso em que o servidor principal e um ou mais dos N servidores falharem (ou suspeitos de estarem falhos).

Vale ressaltar que como a saída fornecida pelo Impact FD não está vinculada ao identificador (ID) dos processos, ele também pode ser aplicado a um contexto em que a composição (*membership*) do sistema não é conhecida ou mesmo em sistemas anônimos (processos não possuem ID) ou homônimos (processos podem ter o mesmo ID).

O restante deste artigo está organizado nas seguintes seções: A Seção 2 apresenta

alguns conceitos básicos sobre detectores de falhas não confiáveis. O Impact FD e alguns exemplos de suas classes são descritos na Seção 3. A Seção 4 resume alguns resultados de avaliação preliminares conduzidos com *traces* reais do PlanetLab [PlanetLab 2014]. A Seção 5 discute alguns trabalhos relacionados. A Seção 6 apresenta a conclusão e os trabalhos futuros.

2. Detector de falhas não confiável

Uma importante abstração para o desenvolvimento de sistemas distribuídos tolerantes a falhas é o detector de falhas não confiável [Chandra and Toueg 1996]. Ele tem por objetivo encapsular a incerteza do atraso de comunicação entre duas entidades distribuídas. Neste trabalho considera-se que existe um processo por nodo (site). Além disso, o termo *processo* pode significar um nodo, um sensor ou um site. Definimos como *correto*, um processo que nunca falha durante toda a execução.

Detectores de falhas não confiáveis são caracterizados por duas propriedades, *completude* (*completeness*) e *acurácia* (*acuracy*), tal como definido em [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.

2.1. Implementação de detectores de falhas

A literatura contém várias propostas de implementação de detectores de falhas não confiáveis que normalmente exploram uma abordagem baseada em tempo (*timer-based*) ou em padrão de mensagem (*message-pattern*).

Na abordagem baseada em tempo, as implementações fazem uso de temporizadores para detectar falhas em processos. Todo processo q envia periodicamente uma mensagem de controle (*heartbeat*) ao processo p responsável pelo monitoramento de q. Se p não receber tal mensagem de q após a expiração de um temporizador, ele acrescenta q à sua lista de processos suspeitos. O uso dos *timeouts* (limite máximo de tempo) pressupõe que o sistema seja síncrono ou, após um tempo, se comporte de maneira síncrona (sincronia parcial) [Chandra and Toueg 1996]. Já a abordagem baseada em padrão de mensagem não usa qualquer mecanismo de tempo limite de espera por uma resposta. [Mostefaoui et al. 2003] propõem uma implementação que faz uso de um mecanismo de requisição-resposta (*query-response*). 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). Uma consulta emitida por p termina quando recebeu α respostas. Os demais processos, cujas respostas não foram consideradas, são vistos como suspeitos de terem falhado. Um processo transmite mensagens $QUERY$ repetidamente se não falhar. Se, na próxima requisição-resposta, p recebe uma resposta de um processo suspeito q, então p remove q de sua lista de suspeitos. Esta abordagem considera uma ordem relativa para a recepção de mensagens em que sempre, ou após um tempo, alguns nodos se comunicam mais rapidamente que outros.

3. Detector de Falhas Impact

Consideramos um sistema distribuído que consiste de um conjunto[1] finito de processos $\Pi = \{q_1, \ldots, q_n\}$, com $|\Pi| = n$. As falhas são apenas por parada (*crash*). Outros tipos de falhas (por exemplo *misbehavior*, transitórias, etc) serão objeto de um trabalho futuro. 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.

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. Portanto, podemos dizer que o sistema é confiável.

Denotamos ${I_p}^S$ o módulo Impact FD do processo p e S é um conjunto de processos de Π. Cada processo $q \in S$ tem um fator de impacto (*impact factor*) associado ($I_q | I_q > 0 : I_q \in \mathbb{R}$). Além disso, o conjunto S pode ser particionado em m subconjuntos disjuntos. Observe que este recurso de agrupamento permite que S seja particionado de acordo com algum critério. Por exemplo, em um cenário em que há diferentes tipos de sensores, aqueles do mesmo tipo podem ser agrupados no mesmo subconjunto. Definimos então $S^* = \{S_1^*, S_2^*, ...S_m^*\}$ como o conjunto S particionado em m disjuntos subconjuntos, onde cada S_i^* é um conjunto composto de tuplas $\langle id, impact \rangle$, sendo id o identificador do processo e $impact$ o valor do fator de impacto do processo.

Quando invocado em p, o Impact FD (${I_p}^S$) retorna o valor ${trust\_level_p}^S$. O ${trust\_level_p}^S$ é um conjunto que contém o nível de confiança (*trust level*) de cada subconjunto S^*, expressando a confiança que p tem no conjunto S. Denotamos ${trusted_p}^S(t) = \{trusted_1(t), \ldots, trusted_m(t)\}$, onde cada $trusted_i(t)$ $(1 \leq i \leq m)$ contém os processos do subconjunto S_i^* que não são considerados falhos por p no tempo $t \in T$. Similarmente ao S_i^*, cada $trusted_i$ é composto da tupla $\langle id, impact \rangle$. A confiança (*trust level*) a $t \in T$ do processo $p \notin F(t)$ com relação ao conjunto S é a função ${trust\_level_p}^S$ tal que ${trust\_level_p}^S(t) = \{trust\_level_i(t) | trust\_level_i(t) = sum(trusted_i(t)); 1 \leq i \leq m\}$. A função $sum(set)$ retorna a soma do fator de impacto de todos os elementos de set.

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 para cada subconjunto, ou seja, é definido como um conjunto que contém o respectivo *threshold* de cada um dos m subconjuntos de S^*: ${threshold}^S = \{threshold_1, \ldots, threshold_m\}$.

O ${threshold}^S$ é usado pela aplicação para verificar a confiança nos processos de S. Se, para cada um dos m subconjuntos de S^*, o $trust\_level_i(t) \geq threshold_i$, 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 (*not trusted*).

[1]Neste trabalho, "conjunto" e "multiconjunto" (um elemento de um multiconjunto pode aparecer mais de uma vez) são usados indistintamente.

$S^* = \{\{\langle q_1, 1\rangle, \langle q_2, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle, \langle q_5, 4\rangle, \langle q_6, 4\rangle\}\}$

t	F(t)	$trusted_p^S(t)$	$trust\_level_p^S(t)$	Status
1	$\{\{\langle q_2, 1\rangle\}, \{\}, \{\}\}$	$\{\{\langle q_1, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle, \langle q_5, 4\rangle, \langle q_6, 4\rangle\}\}$	{1, 3, 12}	TRUSTED
2	$\{\{\langle q_2, 1\rangle\}, \{\}, \{\langle q_6, 4\rangle\}\}$	$\{\{\langle q_1, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle, \langle q_5, 4\rangle\}\}$	{1, 3, 8}	TRUSTED
3	$\{\{\langle q_2, 1\rangle\}, \{\}, \{\langle q_5, 4\rangle, \langle q_6, 4\rangle\}\}$	$\{\{\langle q_1, 1\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_4, 4\rangle\}\}$	{1, 3, 4}	NOT TRUSTED

$threshold^S$ = {1, 3, 8}

Figura 1. Exemplo da saída do detector de falhas

Dois pontos merecem ser ressaltados: (1) ambos o $impactfactor$ 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. Essa capacidade de adaptação é essencial em ambientes dinâmicos, como por exemplo, na computação ubíqua. Observe também que o Impact FD pode ser empregado quando a aplicação precisa de informações sobre cada processo de S individualmente. Neste caso, cada processo deve ser definido como um subconjunto de S^*.

Na Figura 1, consideramos um conjunto S, onde S^* é composto por três subgrupos. Os valores do conjunto $threshold^S$ definem que os subconjuntos S_1 e S_2 devem ter um processo correto, pelo menos, e o subconjunto S_3 deve ter dois. Várias situações são mostradas e o conjunto S é considerado confiável quando, para cada subconjunto S_i^* $(1 \leq i \leq 3)$, o $trust\_level_i(t) \geq threshold_i$.

3.1. Propriedades

Como assinalado na seção anterior, a abordagem do Impact FD possibilita realizar o monitoramento do sistema com maior ou menor rigor. Diferentes classes de Impact FD podem ser definidas, mas que necessitam de diferentes propriedades. Com este objetivo, definimos as seguintes três propriedades.

A propriedade de *flexibilidade* denota a possibilidade do detector fornecer diferentes respostas que levam a um estado confiável sobre o conjunto de processos. Considere X como o conjunto que contém todos os possíveis subconjuntos de elementos que satisfazem o *threshold* definido:

$$X = TPowerSet(S^*, threshold) | TPowerSet(S^*, threshold) = \times\ PowerSet(S_i^*, threshold_i)$$

Inicialmente, a função *TPowerSet* gera o conjunto das partes[2] (ou Power Set) para cada subconjunto (S_i^*) de S^*. Após, são filtrados apenas os subconjuntos de S_i^* cuja a soma dos seus elementos é maior ou igual ao seu *threshold*. Por fim, é feito o produto cartesiano para gerar todas as possibilidades[3]. Assim, X contém todos os subconjuntos possíveis que atendem o $threshold^S$. Vejamos um exemplo:

[2]O conjunto das partes de qualquer S é o conjunto de todos os subconjuntos de S, incluindo o conjunto vazio e o próprio S.

[3]O $\times S_i^*$ é uma forma de abreviar o produto cartesiano quando existem vários conjuntos, por exemplo S_1^*, S_2^*, S_3^* $(X = (S_3^*) \times (S_2^*) \times (S_3^*))$

$S^* = \{\{\langle q_1, 2\rangle, \langle q_2, 4\rangle, \langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}$
threshold^S = {*3, 3*}
$X = TPowerset(S^*, threshold^S)$
PowerSet(S_1^**,* $threshold_1$*)* = $\{\{\langle q_2, 4\rangle\}, \{\langle q_3, 3\rangle\}, \{\langle q_1, 2\rangle, \langle q_2, 4\rangle\}, \{\langle q_1, 2\rangle, \langle q_3, 3\rangle\}, \{\langle q_2, 4\rangle\ \langle q_3, 3\rangle\},$
$\{\langle q_1, 2\rangle, \langle q_2, 4\rangle, \langle q_3, 3\rangle\}\}$
PowerSet(S_2^**,* $threshold_2$*)* = $\{\{\langle q_6, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle\}, \{\langle q_4, 1\rangle, \langle q_6, 3\rangle\}, \{\langle q_5, 2\rangle, \langle q_6, 3\rangle\},$
$\{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}$
X = PowerSet(S_1^**,* $threshold_1$*)* $\times$ *PowerSet(*S_2^**,* $threshold_2$*)*
X = $\{\{\{\langle q_2, 4\rangle\}, \{\langle q_6, 3\rangle\}\}, \{\{\langle q_2, 4\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle\}\}, \{\{\langle q_2, 4\rangle\}, \{\langle q_4, 1\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_2, 4\rangle\},$
$\{\langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_2, 4\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle\}\},$
$\{\{\langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_5, 2\rangle, \langle q_6, 3\rangle\}\}, \{\{\langle q_3, 3\rangle\}, \{\langle q_4, 1\rangle, \langle q_5, 2\rangle, \langle q_6, 3\rangle\}\},$
$\{\{\langle q_1, 2\rangle, \langle q_2, 4\rangle\}, \{\langle q_6, 3\rangle\}\}, \{\{\langle q_1, 2\rangle, \langle q_2, 4\rangle\}, \ldots\}$

Prop. 1 $PR(\diamondsuit IT)$ *(Impact Threshold)*: Para um detector de falhas de um processo p, correto, existe um tempo após o qual o conjunto ${trusted_p}^S$ é um subconjunto de X.

$$PR(\diamondsuit IT) \equiv \exists t \in T, p \in correct(F), \forall t' \geq t, {trusted_p}^S(t') \subset X$$

Prop. 2 ${Impact\ strong\ completeness_p}^S$: Para um detector de falhas de um processo p, correto, existe um tempo após o qual p não confia em qualquer processo falho de S.

$$\exists t \in T, p \in correct(F), \forall q \in (faulty(F) \cap S), \forall t' \geq t, \langle q, \_\rangle \notin {trusted_p}^S(t')$$

Prop. 3 ${Eventual\ impact\ strong\ accuracy_p}^S$: Para um detector de falhas de um processo p, correto, existe um tempo após o qual todos os processos de ${trusted_p}^S$ são corretos.

$$\exists t \in T, \forall t' \geq t, p \in correct(F), \forall \langle q, \_\rangle \in {trusted_p}^S(t'), q \in (correct(F) \cap S)$$

3.2. Exemplos de algumas classes do Impact FD

Assim, quando uma ou mais das propriedades acima descritas são satisfeitas, podemos definir as seguintes classes do Impact FD:

$\diamondsuit$**IT** (*Eventual Impact Threshold Class*): esta classe pode ser implementada quando, considerando nossas definições para o *impact factor* e o *threshold*, o sistema subjacente satisfaz a Prop. 1 (PR ($\diamondsuit IT$)).

Definition 1 ($\diamondsuit IT$): Para um detector de falhas de um processo p, correto, existe um tempo após o qual o ${trust\_level_i}^S(t)$ é maior ou igual ao ${threshold_i}^S$, para todos os subconjuntos de S.

$$\exists t \in T, p \in correct(F), \forall t' \geq t, trust\_level_i(t') \geq threshold_i, \forall 1 \leq i \leq m$$

Na próxima subseção apresentamos uma implementação baseada em *query-response* para a classe $\diamondsuit IT$.

$\diamondsuit$**ICT** (*Eventual Correct Impact Class*): Para o processo p, o conjunto S e um *threshold* definido, as Prop. 1 e Prop. 2 são satisfeitas.

$\diamondsuit$**IPT** (*Eventual Perfect Impact Class*): Para o processo p e o conjunto S, as Prop. 1, Prop. 2 e Prop. 3 são satisfeitas.

3.3. Implementação de um FD da classe $\diamondsuit IT$

O conceito do Impact FD pode ter diferentes implementações de acordo com as premissas do sistema: modelo de sincronismo, o conhecimento ou não do processo p a respeito da composição de S (*membership*), o tipo dos nodos (identificável, anônimo, homônimo), etc. Apresentamos nesta seção um algoritmo que implementa a classe $\diamondsuit IT$ em um sistema assíncrono, baseado no modelo *query-response*. No entanto, vale ressaltar que outra implementação possível seria, por exemplo, com *heartbeat* e *timeout*, considerando um sistema parcialmente síncrono.

Na presente implementação considera-se que a composição do sistema (*membership*) e o número de processo de S não são conhecidos incialmente pelos processos. Além disso, todos os processos se comportam como um nodo de monitoramento, ou seja, $p \in S$. Assume-se que a rede é totalmente conectada, ou seja, $\Lambda = (p, q)|p, q \in \Pi$. Os processos podem invocar a primitava *broadcast(m)* para enviar uma mensagem m para todos os processos do sistema. Se p falha ao transmitir m, esta é recebida por um subconjunto arbitrário de processos. Os links são confiáveis, ou seja, links não perdem, não duplicam, não corrompem, nem geram espúria com relação às mensagens. A propriedade $PR(\diamondsuit IT)$ é satisfeita.

Algoritmo 1 descreve a implementação do $\diamondsuit IT$ FD no processo p em relação à S. Ele se executa por rodadas. A cada nova rodada, caso não falhe, um processo envia por difusão uma mensagem *QUERY*. O intervalo de tempo entre duas rodadas consecutivas é finito mas arbitrário. Quando o processo q recebe uma mensagem *QUERY* de um processo p, q lhe confirma a recepção com uma mensagem *RESP*.

O processo p recebe como entrada seu próprio fator de impacto (I_p), o conjunto a qual pertence ($subset_p$), o *threshold* de cada subconjunto de S^* (conjunto $threshold^S$), o número de subconjuntos de S^* (m), e o número máximo de mensagens que deve esperar (α). Este último é o conjunto $\alpha = \{\alpha_1, \ldots, \alpha_m\}$, onde cada α_i corresponde a um valor limite para o número de mensagens que deve esperar dos processos do subconjunto S_i^*. Por exempo, se f_i denota o número máximo falhas do subconjunto S_i^*, $\alpha_i \leq |S_i^*| - f_i$. No caso do subconjunto de p (que responde à sua própria mensagem *QUERY*), $\alpha_i \leq |S_i^*| - f_i - 1$. A seguinte notação é utilizada:

- r_p: contador de rodadas do processo p;
- *trusted*: é formado por $\{trusted_1, \ldots, trusted_m\}$, onde cada $trusted_i$ ($1 \leq i \leq m$) contém os processos de S_i não considerados falhos por p. Cada $trusted_i$ é um conjunto composto pela tupla $\langle id, impact \rangle$, onde id representa o identificador do processo e $impact$ o valor do fator de impacto do processo que pertence ao subset $trusted_i$;
- *local_known*: conhecimento atual que p tem sobre a composição do sistema (*membership*). É composto de subjconjuntos cujos elementos são do tipo $\langle id, impact \rangle$;
- *trust_level*: conjunto com o nível de confiança de cada subconjunto de processos;
- *X*: conjunto que compreende todos os subconjuntos possíveis formados pelos processos conhecidos por p, cuja soma dos seus elementos é maior ou igual ao *threshold* (veja Seção 3.1). Inicialmente é vazio pois a composição do sistema não é conhecida;
- *sum(set)* retorna a soma do fator de impacto de todos os elementos do conjunto/subconjunto set;
- *size(set)*: função que retorna o número de elementos do conjunto/subconjunto set;

- *Add(set, subset, $\langle q, impact \rangle$)*: função que insere o processo q no subconjunto $subset$ do conjunto set;
- *TPowerSet(set, threshold)*: função que retorna o conjunto que compreende todos os subconjuntos de set cuja a soma dos seus elementos é maior ou igual ao *threshold*.

Algoritmo 1 Algoritmo que implementa a classe $\diamondsuit IT$ FD no processo p

```
 1: Begin
    Input
 2:     I_p, subset_p, threshold^S, m, α

    Init
 3:     r_p ← 0
 4:     local_known ← ∅
 5:     trust_level ← ∅
 6:     X ← ∅
 7:     trusted ← ∅
 8:     c_trusted ← ∅
 9:     Add(local_known, subset_p, ⟨p, I_p⟩)
10:     Add(trusted, subset_p, ⟨p, I_p⟩)
11:     Add(c_trusted, subset_p, ⟨p, I_p⟩)

    Task T1
12:     loop
13:         broadcast(QUERY, r_p)
14:         wait until (size(trusted_i) ≥ α_i |1 ≤
    i ≤ m) or (trusted ⊆ X)
15:         c_trusted ← trusted
16:         trusted ← ∅
17:         Add(trusted, subset_p, ⟨p, I_p⟩)
18:         r_p ← r_p + 1
19:     end loop

    Task T2 - Response
20:     Upon reception of (RESP, r, I_q, subset_q)
    from q do
21:         if r = r_p then
22:             Add(trusted, subset_q, ⟨q, I_q⟩)
23:         end if
24:         if ⟨q, I_q⟩ ∉ susbset_q of local_known
    then
25:             Add(local_known, subset_q, ⟨q, I_q⟩)
26:             X ← TPowerSet(local_known, threshold^S)
27:         end if
28:     end

    Task T3 - Query
29:     Upon reception of (QUERY, r) from q
    do
30:         send(RESP, r, I_p, subset_p) to q
31:     end

    Task T4
32:     Upon invocation of Impact() do
33:         for i = 1 to m do        ▷ for each subset
34:             trust_level_i ← sum(c_trusted_i)
35:         end for
36:         return trust_level
37:     end
38: End
```

O algoritmo possui 4 tarefas. Na tarefa T1 de p há um laço infinito. Primeiro, a mensagem *(QUERY, r_p)* é enviada para todos os processos (linha 13). Em cada rodada (r_p), o processo p espera por pelo menos α_i respostas ($1 \leq i \leq m$) ou até que *trusted* seja um subconjunto de X (isto é, contém os processos que satisfazem o $threshold^S$) (linha 14). Por fim, o contador da rodada (r_p) é incrementado (linha 18).

A tarefa T2 trata a recepção das mensagens *(RESP, r, I_q $subset_q$)* enviadas por q. A mensagem contém a rodada, o fator de impacto e o subconjunto ao qual q pertence. Se a rodada r é igual a r_p, então $\langle q, I_q \rangle$ é adicionado ao conjunto $trusted_q$. Se q ainda não era conhecido por p, então $\langle q, I_q \rangle$ é adicionado a *local_known* no subconjunto $subset_q$ (linha 25). Neste caso, quando um novo processo é adicionado, X é atualizado (linha 26).

A tarefa T3 é responsável pela recepção da mensagem *QUERY*. Quando um processo p recebe uma mensagem *(QUERY, r)* do processo q (linha 29), p deve responder com uma mensagem *RESP* contendo a rodada, o seu fator de impacto e o número do seu subconjunto (linha 30).

A tarefa T4 trata a invocação da função Impact() (linha 32) que retorna o *trust_level* dos processos confiáveis dos subconjuntos de *trusted* (linha 36).

Neste artigo não apresentamos a prova de corretude do algoritmo em função do espaço limitado. Ela pode ser encontrada em [Rossetto et al. 2015].

4. Avaliação de desempenho

Nesta seção descrevemos inicialmente o ambiente no qual os experimentos foram realizados e as métricas de qualidade de serviço (QoS) utilizadas. Em seguida, apresentamos alguns resultados da avaliação com diferentes configurações de conjuntos de nodos no que diz respeito tanto ao fator de impacto quanto ao *threshold*, bem como a comparação com o detector de falhas proposto por [Chen et al. 2002].

4.1. Ambiente

Para os testes de performance, optamos por traces com o mesmo formato que os utilizados para os testes do Accural FD [Hayashibara et al. 2004], sendo que estes consideravam apenas duas máquinas. No nosso caso, a coleta dos traces foi realizada no PlanetLab [PlanetLab 2014] com dez sites, durante uma semana. Cada site enviou mensagens de *heartbeat* para os outros sites a uma taxa de um *heartbeat* a cada 100 ms (intervalo de envio). Sobre os *heartbeats* recebidos pelo site número *1* (considerado o nodo monitor), observamos que os tempos médios entre as chegadas dos *heartbeats* foi muito próximo a 100 ms. No entanto, em alguns sites, o desvio padrão é muito elevado, tal como no site *5* que tem desvio padrão de 310,958 ms com um mínimo de 0,006 ms e um máximo de 657.900,226 ms. Tais valores indicam que, durante um determinado intervalo de tempo da execução, o site parou de enviar *heartbeats* e começou novamente mais tarde. Notou-se também que o site *2* parou de enviar mensagens definitivamente depois de aproximadamente 48 horas. Os demais sites enviaram heartbeats durante todo o periodo de coleta. Finalmente, observamos que os sites *3* e *6* (respec. *5* e *8*) são os sites mais estáveis (respec. instáveis) com desvio padrão de 1,709 ms e 2,557 ms (respec. 310,96 ms e 100,714 ms).

A implementação do Impact FD para a avaliação foi baseada no Algoritmo 1 apresentado em [Rossetto et al. 2015]. Tal algoritmo emprega a abordagem baseada em tempo (*timer-based*) cujo valor do temporizador usa a estimativa de Chen (ver Seção 4.3). Além disso, considera-se um sistema assíncrono, com perda de mensagens em que a propriedade $PR(\diamondsuit IT)$ nao é satisfeita. Devemos ressaltar que, uma vez que os tempos de envio e chegada de cada *heartbeat* são registrados nos arquivos de traços, todos os experimentos foram realizados com exatamente os mesmos cenários e histórico de *heartbeats*.

4.2. Métricas de Qualidade (QoS)

Para avaliar o Impact FD, usamos três das métricas de QoS propostas por [Chen et al. 2002]: *tempo médio de detecção*, *taxa de erros média* e a *probabilidade de uma resposta precisa*. Considerando-se dois processos q e p onde p monitora q, a QoS do detector de falhas em p pode ser determinada a partir das transições entre os estados "trusted" e "not trusted" com relação a q.

- Tempo médio de detecção (T_D): é o tempo que decorre desde o momento em que o processo q falha até que o detector de falhas em p começa a suspeitar de q permanentemente;

Tabela 1. Configurações para o conjunto S

Configuração	Fator de impacto para cada site
Set 0	I_0=2; I_2=1; I_3=6; I_4=6; I_5=1; I_6=6; I_7=1; I_8=2; I_9=2;
Set 1	I_0=1; I_2=2; I_3=6; I_4=6; I_5=2; I_6=6; I_7=2; I_8=1; I_9=1;
Set 2	I_0=6; I_2=2; I_3=1; I_4=1; I_5=2; I_6=1; I_7=2; I_8=6; I_9=6;
Set 3	I_0=2; I_2=6; I_3=1; I_4=1; I_5=2; I_6=1; I_7=2; I_8=6; I_9=6;
Set 4	I_0=2; I_2=1; I_3=6; I_4=6; I_5=1; I_6=6; I_7=2; I_8=2; I_9=1;
Set 5	I_0=3; I_2=3; I_3=3; I_4=3; I_5=3; I_6=3; I_7=3; I_8=3; I_9=3;

- Taxa de erros média (λ_R): representa o número de erros que um detector de falhas faz em uma unidade de tempo, ou seja, a taxa com que o detector de falhas erra;
- Probabilidade de uma resposta precisa (P_A): é a probabilidade de que a saída do detector de falhas esteja correta num dado tempo aleatório.

4.3. Estimativa de chegada de *heartbeats*

Com o intuito de reduzir tanto o número de falsas suspeitas quanto o tempo para detectar uma falha, [Chen et al. 2002] propõem uma abordagem para estimar a chegada do próximo *heartbeat*, que também utilizamos nos nossos experimentos. Para fazer a estimativa, o protocolo se baseia no histórico dos tempos de chegada dos *heartbeats*. Assim, o temporizador é definido de acordo com esta estimativa acrescida de uma margem de segurança (β) constante. A estimativa é recalculada após a chegada de cada novo *heartbeat*. Já a margem de segurança é calculada uma única vez com base nos requisitos de QoS da aplicação.

Os autores sugerem que a margem de segurança β deve variar de 0 a 2500 ms. O tamanho da janela (*window size*) foi definido em 100 amostras para todos os experimentos, o que significa que o detector se baseia apenas nas últimas cem amostras de mensagens de *heartbeat* a fim de calcular a estimativa do tempo de chegada do próximo *heartbeat*.

4.4. Experimentos

Considerando os 10 sites dos traços, definimos que o conjunto S é composto por um único subconjunto com os sites 0, 2, 3, 4, 5, 6, 7, 8 e 9 (*S = {{0, 2, 3, 4, 5, 6, 7, 8, 9 } }*). O site *1* é o monitor (p) e não pertence a S. A Tabela 1 mostra as cinco configurações em relação aos valores do fator de impacto que consideramos para o conjunto S nos experimentos. Para todas as configurações, a soma de fator de impacto dos processos é 27.

As tres métricas propostas por [Chen et al. 2002] e descritas anteriormente foram avaliadas em três diferentes experimentos.

Experimento 1 - Probabilidade de uma resposta precisa: O objetivo deste experimento é avaliar a probabilidade de uma resposta precisa (P_A) com diferentes valores de *threshold* (18, 20, 21, 22, 23, 24 e 25) e diferentes configurações de fator de impacto (Tabela 1). Foi considerada a margem de segurança β = 400 ms.

A Figura 2 mostra que a P_A é menor quando o *threshold* aumenta. É importante lembrar que o *threshold* é um valor limite definido pelo usuário e se o *trust level* fornecido pelo Impact FD é igual ou maior que o *threshold*, a confiança no conjunto de processos

é garantida. Assim, os resultados confirmam que, quando o *threshold* é mais flexível, a probabilidade de uma resposta precisa é maior.

A configuração *"Set 0"* teve a maior P_A para todos os *threshold* devido à atribuição de um alto (respec. baixo) fator de impacto para os sites mais estáveis (respec. instáveis). Por outro lado, as configurações *"Set 2"* e *"Set 3"* tiveram a menor P_A uma vez que os sites instáveis estão com um fator de impacto alto nesses conjuntos. Por exemplo, para o site *8*, a média dos tempos entre as chegadas dos *heartbeats* recebidos é de 100,05 ms, mas o desvio padrão é de 100,71 ms. A atribuição do fator de impacto, portanto, tem influência sobre o desempenho da P_A.

Já a configuração *"Set 5"* apresenta uma queda abrupta da P_A quando o *threshold* é 25. Tal comportamento pode ser explicado uma vez que, nessa configuração de conjunto, todos os sites tem o mesmo fator de impacto (3). Portanto, cada falsa suspeita leva o *trust_level* ser menor que o *threshold* (25), aumentando a duração de erro e, consequentemente, diminuindo a P_A.

Observe, também, que o site *2* falhou após aproximadamente 48 horas. Assim, após a sua falha, a saída do Impact FD, que indica um *trust_level* menor que o *threshold*, não é um erro, ou seja, não é uma falsa suspeita. Por isso, em *"Set 3"*, cujo fator de impacto do site *2* é 6 (alto), a P_A é constante para *threshold* maior ou igual a 22: após a falha do site *2*, o *trust_level* fornecido pelo Impact FD é sempre menor que o *threshold* e falsas suspeitas relacionadas aos outros sites não o modificam. A duração média de erro do experimento é, portanto, menor após a falha, o que melhora a P_A.

Para a comparação da P_A do Impact FD com uma abordagem que monitora os processos individualmente, monitoramos cada site usando o mesmo algoritmo e parâmetros de [Chen et al. 2002] (*WS = 100*; β *= 400 ms*). Com isso, é possível avaliar a capacidade dos dois detectores de falhas (Impact e Chen) em reduzir o número de falsas respostas. A média da P_A obtida foi de 0,979788. Esse resultado mostra que, independentemente da configuração do conjunto, o Impact FD apresenta maior P_A do que [Chen et al. 2002] uma vez que o primeiro tem flexibilidade para tolerar falhas, ou seja, a duração de erro só começa a ser computada quando o *trust_level* fornecido pelo Impact FD é menor que o *threshold* definido, ao contrário do monitoramento individual, onde cada falsa suspeita aumenta a duração de erro.

Os resultados deste experimento destacam que a atribuição de diferentes valores de fator de impacto para os nodos pode degradar o desempenho do detector de falhas, especialmente quando os sites instáveis tem alto fator de impacto.

Experimento 2 - Tempo médio de detecção: No segundo experimento foi avaliada a probabilidade de uma resposta precisa (P_A) em relação ao tempo médio de detecção (T_D). Para este fim, variamos a margem de segurança (β) para obter diferentes valores de tempo de detecção. Esta então variou com intervalos de 100 ms, a partir de 100 ms. Neste experimento usamos a configuração *"Set 0"* e definimos diferentes *threshold*. Esta configuração de conjunto foi escolhida porque apresentou a melhor P_A para todos os *threshold* no Experimento 1. Também avaliamos a P_A e o T_D para o algoritmo de [Chen et al. 2002], o qual fornece como saída o conjunto de nodos suspeitos.

A Figura 3 mostra que para um *threshold* alto e tempo médio de detecção próximo a 200 ms, a P_A do Impact FD é menor, independentemente do *threshold*, isto porque a

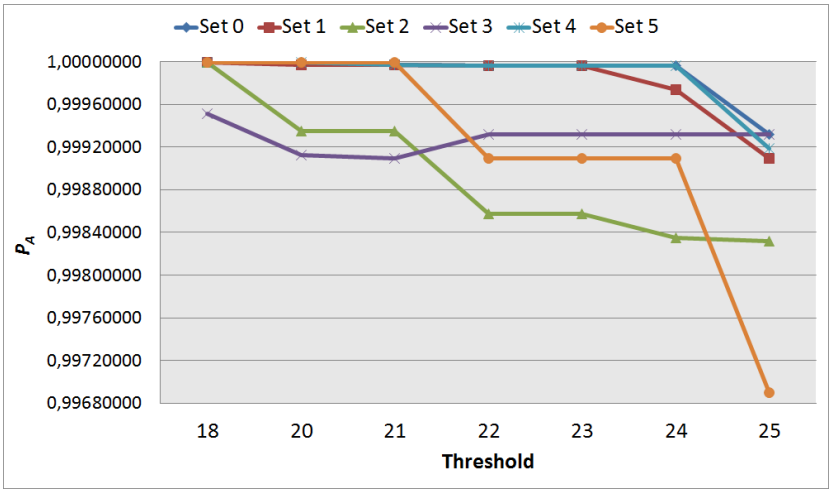

Figura 2. P_A **X threshold com diferentes configurações para o conjunto *S***

margem de segurança é, neste caso, igual a 100 ms, o que aumenta ambos o número de falsas suspeitas e a duração de erro. No entanto, quando o T_D é maior que 230 ms, a P_A do Impact FD é consideravelmente maior do que Chen. Após o tempo de detecção de aproximadamente 400 ms, a P_A do Impact FD torna-se constante, independentemente do tempo de detecção e *threshold*, ficando próxima a 1. Tal comportamento pode ser explicado, pois quanto maior a margem de segurança, menor é o número de falsas suspeitas e a duração de erro, o que confirma que quando o *timeout* é curto, as falhas são detectadas mais rapidamente, mas a probabilidade de ter falsas detecções aumenta [Satzger et al. 2007].

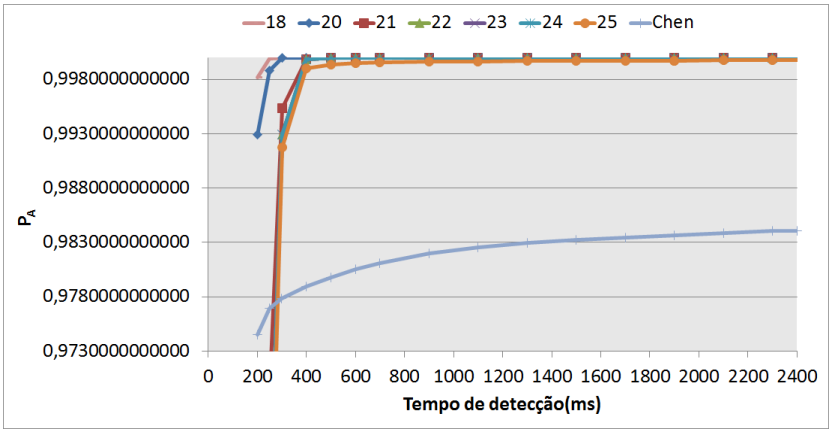

Figura 3. P_A **X** T_D **com diferentes *thresholds***

Experimento 3 - Taxa de erros média: Neste experimento avaliamos o tempo médio de detecção versus a taxa de erros média (erros por segundo). Consideramos a configuração *"Set 0"* e a taxa de erros é expressa em uma escala logarítmica. Podemos observar na Figura 4 que a taxa de erros média do Impact FD é maior quando o tempo de detecção é baixo (menor que 400 ms) e o *threshold* é alto (23-25). Tal resultado está de acordo com Experimento 2: sempre que a margem de segurança é pequena e o *threshold* tolera menos falhas, o Impact FD comete erros com mais freqüência. Em outras palavras, a taxa de erro diminui quando o *threshold* é mais flexível ou o tempo de detecção aumenta.

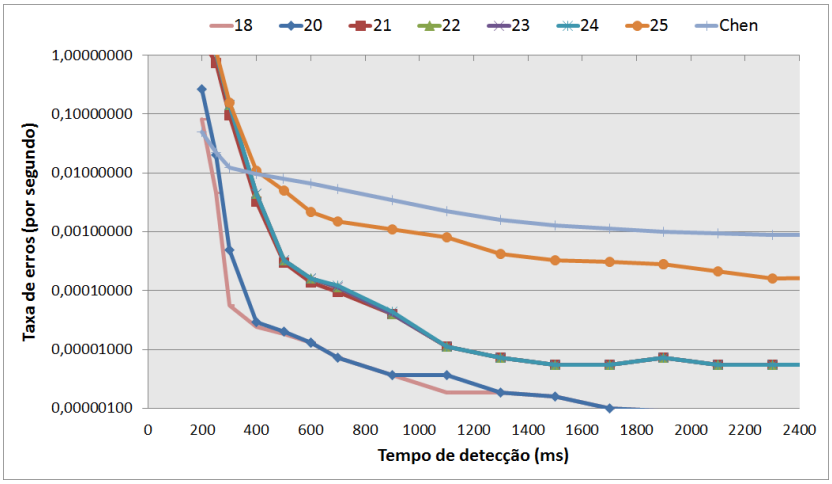

Figura 4. λ_R X T_D com diferentes *thresholds*

5. Trabalhos Relacionados

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]). Porém, em alguns detectores, como por exemplo os da classe Σ (respec. Ω) [Delporte-Gallet et al. 2004], o conjunto de processos (respec., o processo) de saída refere-se aos processos não suspeitos de estarem falhos ($trusted$).

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] é um trabalho recente que 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*).

6. Conclusão e Trabalhos Futuros

Neste artigo, apresentamos e definimos o detector de falhas não confiável Impact FD, que fornece uma saída relacionada com um conjunto de processos e não apenas para cada um individualmente. Ambos o *fator de impacto* e o *threshold* oferecem um grau de flexibilidade já que permitem ao usuário ajustar o Impact FD de acordo com as necessidades específicas e a margem de falhas aceitável da aplicação. Em alguns cenários e configurações, eles também podem reduzir a taxa de falsas respostas quando comparado com detectores de falhas não confiáveis tradicionais. Os resultados da avaliação de desempenho mostram que a atribuição de um alto (respec. baixo) fator de impacto para os nodos mais estáveis (respec. instáveis) aumenta a probabilidade de uma resposta precisa do detector de falhas.

Como trabalho futuro, pretende-se estender o Impact FD a fim de abordar as falhas de comportamento impróprio (*misbehavior*). Também estamos trabalhando na redução e equivalência do Impact FD em relação a outros detectores (por exemplo, Sigma e Ômega [Delporte-Gallet et al. 2004]), o que torna necessário algumas condições e/ou domínios específicos como hipótese. Outra direcão de nossa pesquisa é a realização de outros experimentos em redes diferentes, tais como Wi-Fi ou rede local.

Referências

Bertier, M., Marin, O., Sens, P., et al. (2003). Performance analysis of a hierarchical failure detector. In *DSN*, volume 3, pages 635–644.

Brun, Y., Edwards, G., Bang, J. Y., and Medvidovic, N. (2011). Smart redundancy for distributed computation. In *ICDCS)*, pages 665–676.

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

Chen, W., Toueg, S., and Aguilera, M. K. (2002). On the quality of service of failure detectors. *Computers, IEEE Transactions on*, 51(5):561–580.

Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Hadzilacos, V., Kouznetsov, P., and Toueg, S. (2004). The weakest failure detectors to solve certain fundamental problems in distributed computing. In *ACM PODC*, pages 338–346.

Hayashibara, N., Defago, X., Yared, R., and Katayama, T. (2004). The φ accrual failure detector. In *SRDS*, pages 66–78.

Leners, J. B., Gupta, T., Aguilera, M. K., and Walfish, M. (2013). Improving availability in distributed systems with failure informers. In *NSDI*.

Mostefaoui, A., Mourgaya, E., and Raynal, M. (2003). Asynchronous implementation of failure detectors. In *DSN*, pages 351–351.

PlanetLab (2014). Planetlab. `http://www.planet-lab.org`.

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

Satzger, B., Pietzowski, A., Trumler, W., and Ungerer, T. (2007). A new adaptive accrual failure detector for dependable distributed systems. In *ACM SAC*, pages 551–555.