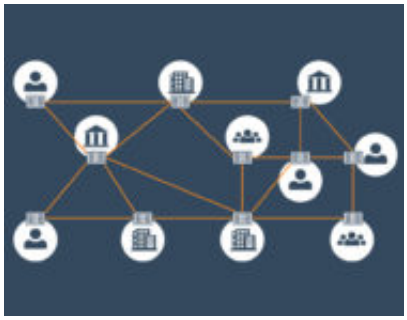
Distributed Ledger: una tecnologia abilitante per la resilienza ICT

Author : Fabrizio Baiardi

Date : 6 Marzo 2019



Dopo aver discusso alcune funzioni per aumentare la resilienza di un sistema ICT, consideriamo lo scambio di informazioni fra i moduli che implementano queste funzioni. I moduli possono operare correttamente solo se possono scambiare informazioni corrette e consistenti. Proponiamo di implementare queste comunicazioni attraverso un ledger distribuito. Il meccanismo del consenso alla base di un ledger può garantire una corretta trasmissione di informazioni anche se alcuni nodi sono maliziosi o guasti. Dopo aver descritto la soluzione proposta, presentiamo una prima analisi delle prestazioni.

1. Introduzione

La diffusione crescente di sistemi ICT nella nostra società sottolinea l'importanza di come essi debbano poter tollerare e reagire all'occorrenza di minacce che riducono le loro funzionalità e i servizi che offrono [1], [2], [3]. In principio, esistono diverse prospettive per analizzare il problema, in base al tipo di evento, per esempio se naturale o malizioso, la quantità di informazioni che sono disponibili sull'evento prima del suo avvenimento e le conseguenze attese. Ad esempio, le analisi di safety e di security si concentrano, rispettivamente, su eventi naturali e sul comportamento malizioso di qualche attaccante. Un framework sulla resilienza fonde le due prospettive per coprire più sorgenti di attacco.

Esistono definizioni alternative di resilienza. Ad esempio, "La resilienza è l'abilità di una architettura di supportare le funzioni necessarie per il successo della missione a dispetto di azioni ostili o condizioni avverse. Un'architettura è più resiliente se può fornire queste funzioni con più alta probabilità, periodi più brevi di capacità ridotta, e attraverso una più ampia gamma di scenari, condizioni e minacce"[4]. Più in generale, la resilienza si riferisce all'abilità di prepararsi e adattarsi a condizioni mutevoli e resistere e recuperare rapidamente dai disturbi di tipi di attacchi deliberati, eventi accidentali, minacce naturali o incidenti. Nel caso di infrastrutture informatiche, la **cyber resilienza** indica la capacità di un sistema di recuperare o di rigenerare le sue prestazioni dopo un degrado dovuto a guasti o ad attacchi.

In un articolo precedente abbiamo visto che la resilienza di un sistema può essere ottenuta integrando ridondanza, monitoraggio e riconfigurazione. In maggior dettaglio, possiamo ottenere la cyber resilienza integrando **cinque funzioni:**

- Anticipazione: scoprire e mitigare le vulnerabilità,
- Rilevamento: scoprire attacchi in corso, crash e fallimenti,
- Confinamento: cambiare la superficie d'attacco corrente per minimizzare la probabilità di successo di attacchi in corso o isolare i moduli che l'attaccante controlla,
- Ripristino: ripristinare le funzionalità originali del sistema quando l'attacco termina o parziali durante l'attacco,
- Miglioramento: aggiornare il sistema per migliorare la resilienza.

Questo articolo discute lo scambio di informazioni tra queste funzioni. In particolare, discute come le componenti che implementano il rilevamento possano interagire in modo sicuro con quelle di confinamento o di ripristino. Queste interazioni sono **altamente critiche** perché determinano le azioni di confinamento e di ripristino per ripristinare alcune funzionalità del sistema, ad esempio alcuni dei servizi offerti. Quindi, la trasmissione di false informazioni sugli attacchi in corso o sullo stato di alcune componenti risulterà in un comportamento inconsistente dei moduli che realizzano il confinamento o il ripristino. Questo è il motivo per cui proponiamo di implementare lo scambio di informazioni fra il task di rilevamento e quelli di confinamento e ripristino attraverso un ledger distribuito [5], [6], [7], una struttura dati che può superare i fallimenti bizantini [8], [9], [10] dei moduli che aggiornano il ledger.

Il cap. 2 di quest'articolo richiama brevemente i task principali che un sistema deve implementare per ottenere la resilienza. In seguito, il cap. 3 descrive come un ledger distribuito possa offrire un canale di comunicazione sicuro ed affidabile che può tollerare errori ed attacchi. Il cap. 4 mostra come i moduli che monitorano un sistema possano usare questo canale per trasmettere informazioni a quelli che riconfigurano il sistema. Il cap. 5, infine, discute una possibile implementazione e una prima valutazione delle prestazioni.

2. Funzioni per la cyber resilienza

Oltre ai servizi normali, un cyber-resiliente [11], [12] deve eseguire cinque funzioni: anticipazione, rilevamento, riconfigurazione, ripristino e miglioramento. I moduli che realizzano ognuna delle funzioni e quelli che si occupano delle attività normali lavorano concorrentemente, cioè alcuni moduli possono essere coinvolti in attività normali ed altri in una riconfigurazione. Come discusso di seguito, ogni funzione può essere implementata in parallelo da moduli distinti con un certo grado di ridondanza. Introduciamo brevemente le cinque funzioni.

La funzione di *anticipazione* scopre le vulnerabilità correnti nelle componenti del sistema e le mitiga prima che un evento malizioso o naturale le sfrutti. Questa funzione comprende, tra gli altri, il *vulnerability scanning* e l'installazione di patch o di una nuova versione di un modulo.

La funzione di *rilevamento* monitora continuamente il sistema per scoprire e correlare gli attacchi in corso e il loro obiettivo. Le tecnologie a supporto includono il rilevamento *signature-based* e *anomaly-based* [13]. La funzione si occupa sia di guasti che di incidenti da attacchi

maliziosi. Un guasto è un evento singolo che è solitamente rilevato usando il time out così come implementato nel *timer watchdog*; un attacco malizioso è formato da una catena di attacchi mirati a incrementare i privilegi dell'attaccante. Ogni attacco nella catena aumenta i privilegi. Proprio perché deve fronteggiare una catena di attacco, il sistema può rilevare e reagire alla catena prima che un attaccante la esegua completamente. Di seguito, quando si discute di un attaccante intelligente, usiamo il termine attacco per denotare un'intera catena di attacco mentre denotiamo come attacco elementare un elemento della catena. Il *rilevamento* correla l'attacco elementare corrente con quelli già rilevati per scoprire l'intera catena e predirne l'obiettivo finale Inoltre, esso deve definire lo stato, e.g. attaccato, integro o guasto, dei moduli del sistema. Questa informazione è fondamentale per riconfigurare il sistema e continuare ad offrire dei servizi.

Il *confinamento* opera a partire dai risultati del *rilevamento*. Usando le informazioni sulle catene di attacco e la previsione degli obiettivi finali, il *confinamento* decide se e come riconfigurare il sistema per resistere meglio agli attaccanti e continuare ad offrire dei servizi. Un sistema resiliente può sospendere alcuni servizi quando o non può implementarli in modo affidabile, o potrebbero essere il prossimo passo nella catena perché appartengono all'attuale superficie d'attacco. Questa superficie dipende dai privilegi che l'attaccante ha acquisito attraverso gli attacchi precedenti. Un modulo può essere sospeso spegnendolo, o riconfigurando la topologia del sistema in modo tale che non possa più essere raggiunto.

Il *ripristino* riporta il sistema ad uno stato normale, o almeno ad uno in cui alcuni servizi sono disponibili. Ha inizio dopo un intervallo di tempo predefinito dall'ultimo attacco di un attaccante malizioso o dopo un evento naturale. Opera a partire dallo stato aggiornato dei moduli del sistema che riceve dal rilevamento e riattiva alcune componenti e/o riconfigura le interconnessioni fra componenti. **Se gli attacchi hanno un impatto fisico** sulle componenti del sistema, un sistema completamente funzionale può essere ripristinato solo se il sistema presenta una qualche ridondanza fisica. Di seguito, assumiamo che la riconfigurazione possa anche avvenire attraverso alcuni piani pre-calcolati. Questo implica che confinamento e riconfigurazione richiedono solo la scoperta dei i moduli affetti da guasti o attacchi.

Le prime quattro funzioni gestiscono un crash e/o un fallimento finché non è risolto o l'attaccante è sconfitto. Il *miglioramento* deve ottimizzare i vari parametri e le soglie che guidano il comportamento delle altre funzioni per minimizzare la perdita di prestazioni. Non usiamo il termine apprendimento perché questa funzione può non adottare metodologie di apprendimento standard.

L'**obiettivo complessivo** delle varie funzioni è che il sistema sopravviva agli attacchi, offra dei servizi nonostante l'attacco e ritorni alle sue piene funzionalità il prima possibile. Di seguito discutiamo come garantire la consistenza e l'integrità delle comunicazioni fra rilevamento, confinamento e ripristino. Questi attributi sono importanti perché le funzioni possono essere efficaci solo se dispongono di informazioni accurate e consistenti sullo stato del sistema.

Comunicazioni resilienti attraverso un distributed ledger

Dopo aver presentato brevemente un distributed ledger, discutiamo di come questa struttura

possa supportare alcune comunicazioni in un sistema resiliente.

2.1 Distributed Ledger

Un distributed ledger [14], [15] [16] è un **database di asset** che può essere condiviso su una rete di più siti, geografie o istituzioni. Tutti i partecipanti all'interno di una rete possono avere la loro copia identica del ledger perché tutti i cambiamenti al ledger vengono riflessi in tutte le copie. Le imprese usano questa tecnologia per processare, validare o autenticare transazioni o altri tipi di scambio dati. Tipicamente, questi record sono salvati nel ledger solo quando è stato raggiunto il consenso fra tutte le parti coinvolte. A ogni record nel ledger vengono associati un timestamp e una firma crittografica univoca. La tecnologia fornisce una storia verificabile ed ispezionabile di tutte le informazioni nel dataset che un ledger memorizza e tutti i partecipanti nella rete possono vedere tutta la storia.

L'aggiornamento di un ledger distribuito adotta un **algoritmo di consenso** per determinare l'informazione da inserire per estenderlo. Alcuni algoritmi sono basati su una *proof of work*, cioè il modulo che può inserire un record è il primo che risolve un problema che richiede un pesante lavoro computazionale. In questo caso, non abbiamo nessun consenso a priori ma solo un meccanismo di arbitraggio tra le varie richieste. Esiste un meccanismo di consenso a posteriori perché, se un modulo malizioso inserisce un record con informazioni false nella catena, quelli corretti la trascureranno e continueranno la catena dal record precedente. In un ledger, questa fork può produrre una situazione in cui due o più record hanno la stessa altezza, cioè un record ha più di un successore. La fork viene risolta quando i moduli corretti aggiungono altri record ed una delle catene diventa più lunga rispetto alle alternative. I moduli corretti abbandonano i record al di fuori della catena più lunga e che diventano record *orfani*.

Altre soluzioni raggiungono il consenso a priori attraverso alcuni round di voto finché una percentuale predefinita dei partecipanti della rete si accorda sullo stesso record. Di seguito, consideriamo un **problema** in cui tutti i moduli che possono accedere ed aggiornare il ledger distribuito sono noti in anticipo. Questo aumenta il numero dei possibili algoritmi di consenso.

2.2 Un distributed ledger come canale

Consideriamo un'applicazione del ledger distribuito diversa dalle tradizionali perché nella maggior parte dei sistemi un ledger sostituisce un database centralizzato o distribuito [17], [18], [19], [20]. Ad esempio, un sistema per una supply chain memorizza la provenienza di ogni componente usato. In questo caso si sostituisce un sistema di database centralizzato con un distributed ledger in cui ogni fornitore gestisce una copia del ledger. Il distributed ledger opera come **sorgente sincronizzata** dei dati che sono successivamente distribuiti alle altre copie. In questo modo, ogni fornitore non gestisce un solo nodo della rete ma replica, valida e gestisce un ledger. Ogni nodo della rete propone aggiornamenti e determina quelli da eseguire sulle copie del ledger. In questa soluzione, il ledger coordina i sistemi ICT dell'organizzazione nella rete.

La soluzione proposta utilizza un ledger distribuito come canale di comunicazione fra moduli di

un sistema. Anche se le proprietà teoriche non dipendono dimensioni del ledger, ci sono alcune differenze. Ad esempio, la frequenza di inserimento è molto più elevata in una struttura dati.

La comunicazione mediante ledger può resistere sia a guasti sia ad attacchi maliziosi ai moduli che aggiornano e accedono il ledger. Consideriamo, ad esempio, un **problema produttore/consumatore** con molti produttori e molti consumatori. I produttori calcolano delle informazioni, mentre i moduli consumatori utilizzano le informazioni prodotte. Consumatori distinti possono consumare la stessa informazione. I produttori salvano le informazioni che calcolano in un ledger distribuito da cui ogni consumatore preleva i suoi input. Per semplicità, assumiamo che ogni produttore possieda una copia locale del ledger e che tutti i produttori decidano le informazioni da inserire nel record successivo del ledger mediante un algoritmo di consenso. Assumiamo anche che il ledger consista di una catena di record e che ognuno ricordi lo stato dei moduli di interesse a un certo istante. L'intero ledger memorizza la storia dei moduli di interesse. I produttori corretti, quelli non guasti e non controllati dell'attaccante, calcolano un nuovo record solo dopo aver aggiunto il precedente al ledger. Questa sincronizzazione è garantita dall'algoritmo di consenso che permette ai produttori di calcolare l'informazione nell'i+1-esimo record del ledger solo dopo aver raggiunto il consenso sull'i-esimo.

Alcuni dei produttori possono essere maliziosi e/o guasti e quindi tentano di inserire informazioni false nel ledger con l'obiettivo di provocare decisioni inconsistenti nel consumatore. Nonostante questi moduli, le informazioni nel ledger sono affidabili perché **l'algoritmo del consenso garantisce che le informazioni che vengono salvate nel ledger dipendano solo dai produttori corretti**. Questo può essere garantito ogni volta che il numero di produttori maliziosi è inferiore a una data percentuale del numero totale di produttori. Ad esempio, alcuni ledger adottano un algoritmo del consenso *"pratical byzantine fault tolerant"* [21], [22], [23] che garantisce che i consumatori corretti raggiungano il consenso sul valore corretto purchè il numero n di produttori e quello f di produttori guasti o maliziosi soddisfino $n > 5f + 1$. Questo implica anche che almeno $f + 1$ produttori calcolino lo stato di ogni nodo. Sotto questa condizione, possiamo assumere che le informazioni nel ledger di un produttore non guasto o malizioso sia sempre consistente.

L'algoritmo di consenso garantisce che il ledger memorizzi solo informazioni affidabili purché esista un'intersezione fra gli output di produttori diversi, cioè più produttori calcolino lo stesso output. Di conseguenza, l'output di un produttore permette di controllare la consistenza di quello di altri produttori. Supponiamo, ad esempio, che le informazioni di interesse descrivano lo stato attaccato/sicuro di ognuno di un insieme di moduli. In questo caso, ogni produttore calcola e salva nel ledger lo stato di alcuni moduli e più produttori calcolano e salvano lo stato di ogni modulo. L'informazione ridondante su ogni modulo permette di controllare la consistenza dell'output di ogni produttore. Assumiamo anche che il ledger consista di una catena di record e che ognuno ricordi lo stato dei moduli di interesse a un certo istante. L'intero ledger memorizza la storia dei moduli di interesse. I produttori corretti, cioè quelli che non sono guasti e non sotto il controllo dell'attaccante, sono sincronizzati in modo da calcolare un nuovo record solo dopo aver aggiunto il precedente al ledger. Questo vincolo è soddisfatto dal meccanismo del consenso che permette ai produttori di calcolare l'informazione nel record i+1-esimo del ledger solo dopo aver raggiunto il consenso sull'i-esimo.

I consumatori recuperano le informazioni di interesse accedendo al ledger. Ad esempio, possono leggere il numero di nodi guasti o attaccati e azionare alcune azioni per riconfigurare il sistema. La lettura delle informazioni nel ledger deve considerare che un produttore malizioso può trascurare l'algoritmo di consenso ed aggiornare la sua copia del ledger usando informazione falsa. Quindi, **un consumatore deve confrontare più copie del ledger** e può fidarsi delle informazioni nel ledger solo se almeno *f + 1* copie distinte del ledger concordano sullo stesso valore. Ad esempio, un consumatore è certo che un modulo è stato attaccato se almeno *f*+1 copie del ledger memorizzano questo stato del modulo. Di seguito, denotiamo come validati, i dati salvati in almeno *f*+1 ledger distinti. Il meccanismo di validazione richiede semplicemente di leggere le stesse informazioni in più copie del ledger.

3. Attivare una riconfigurazione

Nella progettazione di un sistema resiliente, il problema produttore/consumatore più interessante sorge **fra i monitor e i manager**, i moduli che implementano, rispettivamente, il rilevamento e confinamento e la riconfigurazione.

Consideriamo per primi i produttori, cioè i monitor. Questi moduli valutano, mediante opportuni test, lo stato degli altri moduli per rilevare se ciascuno è sicuro, è guasto o è stato attaccato con successo. La relazione fra questi produttori e i moduli del sistema è molti-a-molti perché monitor diversi controllano lo stesso modulo mentre lo stesso monito controlla molti moduli. Come già detto, se al più *f* monitor possono essere guasti o maliziosi, almeno *f+1* monitor dovrebbero controllare ogni modulo per garantire che almeno uno di loro non sia guasto.

In un sistema resiliente, i consumatori sono i manager che aggiornano la configurazione di sistema per confinare i moduli guasti o attaccati con successo. I manager possono migrare i moduli fra i nodi del sistema per ripristinare alcuni servizi. La tecnologia principale sottostante la riconfigurazione e il confinamento è la **virtualizzazione** [24], una tecnologia chiave per la resilienza perché disaccoppia completamente l'implementazione di un ambiente dalle risorse hardware. Una macchina virtuale è un ambiente di programmazione autonomo, auto contenuto con sistema operativo, applicazioni e server. È mappata su un nodo fisico e può migrare su un nodo diverso in modo completamente trasparente per le sue applicazioni e i suoi utenti. Inoltre, il sistema può facilmente duplicare una macchina virtuale per motivi di ridondanza o per creare un backup economico con tutti i dati e il codice della macchina. La memorizzazione del backup su un nodo diverso è il meccanismo di base di riconfigurazione e ripristino di un servizio quando quello originale è guasto o è stato attaccato.

La **migrazione** di una macchina virtuale è anche un **meccanismo di difesa** poiché offusca il nodo fisico da attaccare per raggiungere una certa macchina virtuale. Ad esempio, l'anticipazione può implementare strategie di inganno attive che migrano ogni macchina su un nodo selezionato casualmente con una frequenza predefinita. Questo incrementa fortemente la complessità di attacchi che eseguono del software sullo stesso nodo della macchina virtuale di interesse. Un esempio sono gli attacchi side-channel, che rubano informazioni da una macchina sfruttando il monitoraggio dell'uso delle risorse. La virtualizzazione semplifica anche l'implementazione di honeynet altamente interattive che non solo rallentano un attaccante ma collezionano anche informazioni sul suo conto. Infine, la virtualizzazione supporta

l’introspezione di macchine virtuali, una tecnica potente per la scoperta di malware mediante analisi della memoria.

Nella **soluzione proposta**, i manager recuperano in un ledger distribuito, quello dello stato, l’informazione per determinare le macchine virtuali da ripristinare. I manager possono anche aggiornare la tabella di routing dei messaggi nei nodi del sistema per isolare un modulo.

Una **questione critica** è come evitare che un manager guasto o sotto il controllo dell’attaccante invalidi i due task. Anche in questo caso, implementiamo con un ledger distribuito, quello dell’azione, la comunicazione fra i monitor e i supervisori del nodo, i moduli che implementano la riconfigurazione in ciascun nodo. Nel problema considerato, ogni monitor produce le informazioni nel ledger mentre ogni supervisore del nodo le consuma. Anche in questo caso, un supervisore aggiorna la configurazione del proprio nodo, cioè le macchine virtuali e la tabella di routing, solo dopo aver validato l’informazione sullo stato. Il meccanismo di consenso e quello di validazione garantiscono la consistenza dell’input di ogni supervisore.

Si noti che sia i ledger di stato che quelli di azione - che implementano la comunicazione fra, rispettivamente, i monitor e i manager e i manager e i supervisori dei nodi - sono una catena di record. L’ultimo record nel ledger di stato memorizza informazioni riguardo allo stato attuale dei moduli del sistema, mentre l’ultimo record in quello di azione codifica l’azione da eseguire per ripristinare le prestazioni del sistema.

Possiamo evitare i manager e il ledger che supporta la comunicazione tra manager e supervisori solo se le azioni di ogni supervisore dipendono unicamente dallo stato degli altri moduli. Ad esempio, il comportamento del supervisore del nodo *c* può essere guidato dalla **regola:**

> *Se i nodi a e b sono down, allora ferma le macchine virtuali Va, Vb e Vc e riattiva Vx e Vy.*

Nell’esempio, Vx e Vy possono essere copie di macchine in esecuzione sul nodo a o sul nodo b. Questa policy di riconfigurazione è statica e quindi più efficiente ma può ridurre la resilienza globale, perché decide in anticipo le possibili reazioni.

Le **fig. 1-3** riportano alcune prestazioni di un distributed ledger in cui alcuni nodi sono maliziosi. Il punto interessante è la forte correlazione delle prestazioni con la frequenza di transazioni maliziose, indipendentemente dal numero di nodi maliziosi.

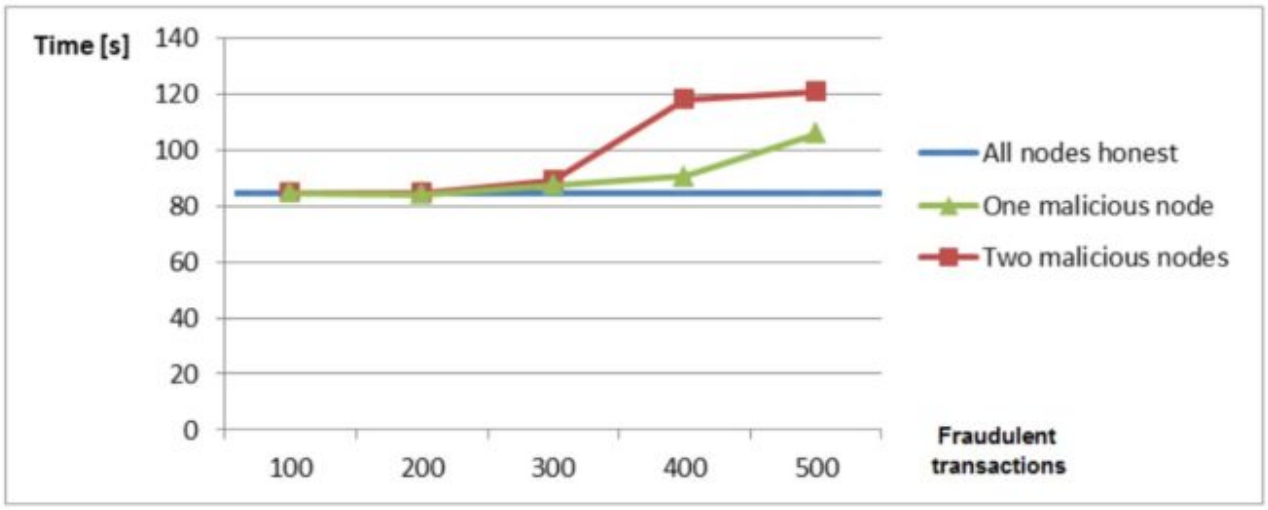


Fig. 1: Degradazioni delle Prestazioni con un Nodo Malizioso, quorum al 60%

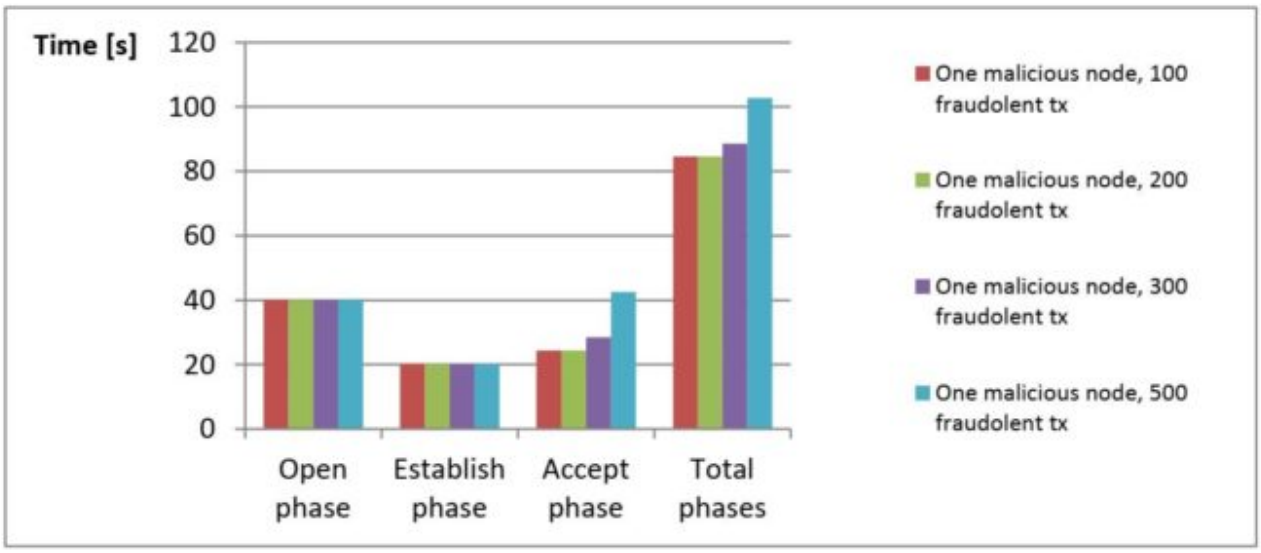


Fig. 2: Degradazione delle Prestazioni con un Nodo Malizioso, quorum all' 80%

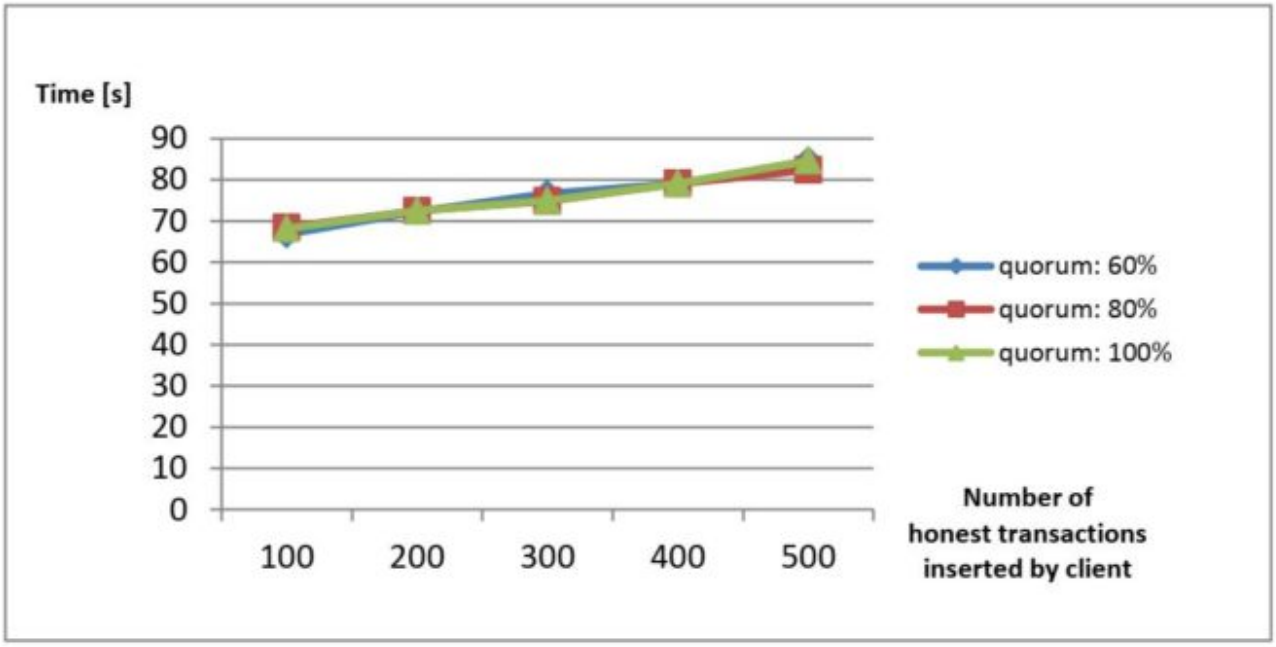


Fig. 3: Degradazione delle Prestazioni per valori diversi del quorum

Note

[1] D J. Bodeau, D. Graubart, J. Picciotto, R. McQuaid, *Cyber Resiliency Engineering Framework*, Technical rep. R110237, Sept. 2011, MITRE Corporation.

[2] B. Donnet and T. Friedman, *Internet topology discovery: a survey*, IEEE Communications Surveys & Tutorials, vol. 9, no. 4, Fourth Quarter 2007, pp. 56-69.

[3] NIAC, *A Framework for Establishing Critical Infrastructure Resilience Goals: Final Report and Recommendations by the Council*, NIAC, October 19, 2010.

[4] NIAC, A Framework for Establishing Critical Infrastructure Resilience Goals: Final Report and Recommendations by the Council, NIAC, October 19, 2010.

[5] Gov. Office for Science, Uk, *Distributed Ledger Technology: beyond block chain: A report by the UK Government Chief Scientific Adviser*, 2016.

[6] K. Wüst and A. Gervais, *Do you Need a Blockchain?*, 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), Zug, 2018, pp. 45-54. doi: 10.1109/CVCBT.2018.00011

[7] Z. Zheng, et al. *Blockchain challenges and opportunities: A survey*, *International Journal of Web and Grid Services* 14.4 (2018): 352-375.

[8] M. J. Fischer, N. A. Lynch and M.S. Paterson, *Impossibility of Distributed Consensus with One Faulty Process*, J. ACM, vol. 32, 1985, pp. 374-382.

[9] Y. Gilad, R. Hemo, S. Micali, G. Vlachos and N. Zeldovich, *Algorand: Scaling Byzantine Agreements for Cryptocurrencies*, IACR Cryptology ePrint Archive, 2017.

[10] S. Joao, A. Bessani, and M. Vukolic, *A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain Platform, 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018.

[11] D J. Bodeau, D. Graubart, J. Picciotto, R. McQuaid, *Cyber Resiliency Engineering Framework*, Technical rep. R110237, Sept. 2011, MITRE Corporation.

[12] S. Nakamoto, *Bitcoin: a Peer-to-Peer Electronic Cash System*, 2008.

[13] S. Joao, A. Bessani and M. Vukolic, *A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform*, *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018.

[14] Gov. Office for Science, Uk, *Distributed Ledger Technology: beyond block chain: A report by the UK Government Chief Scientific Adviser*, 2016.

[15] K. Wüst and A. Gervais, *Do you Need a Blockchain?*, 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), Zug, 2018, pp. 45-54. doi: 10.1109/CVCBT.2018.00011.

[16] Z. Zheng, et al. *Blockchain challenges and opportunities: A survey*, *International Journal of Web and Grid Services* 14.4 (2018): 352-375.

[17] Gov. Office for Science, Uk, *Distributed Ledger Technology: beyond block chain: A report by the UK Government Chief Scientific Adviser*, 2016.

[18] S. Joao, A. Bessani, and M. Vukolic. "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform", *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018.

[19] S. Joao, A. Bessani, and M. Vukolic, *A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform*, *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018.

[20] K. Wüst and A. Gervais, *Do you Need a Blockchain?*, 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), Zug, 2018, pp. 45-54. doi: 10.1109/CVCBT.2018.00011.

[21] B. Chase and E. MacBrough, *Analysis of the XRP Ledger Consensus Protocol,* 2018.

[22] Y. Gilad, R. Hemo, S. Micali, G. Vlachos and N. Zeldovich, *Algorand: Scaling Byzantine Agreements for Cryptocurrencies*, IACR Cryptology ePrint Archive, 2017.

[23] NIAC, *A Framework for Establishing Critical Infrastructure Resilience Goals: Final Report and Recommendations by the Council*, NIAC, October 19, 2010.

[24] A. Wang, M. Iyer, R. Dutta, G.N. Rouskas, G. N., and I. Baldine, *Network virtualization: Technologies, perspectives, and frontiers,* Journal of Lightwave Technology, 31(4), 2013, 523-537.

Articolo a cura di **Fabrizio Baiardi** e **Francesco Balzano**, Università di Pisa