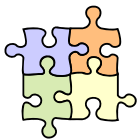


Tattiche architetturali
(seconda parte)

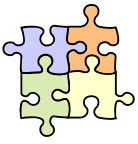
Dispensa ASW 311

ottobre 2011



- Fonti

- [SAP] Chapter 5, Achieving Qualities
- [Bachmann, Bass, Nord, 2007] Modifiability Tactics, Technical report CMU/SEI-2007-TR-002
- [Scott, Kazman, 2009] Realizing and Refining Architectural Tactics: Availability, Technical report CMU/SEI-2009-TR-006
- [Kim, Kim, Lu, Park, 2009] Quality-driven architecture development using architectural tactics, The Journal of Systems and Software, 82, 2009
- [Parnas, 1972] On the Criteria To Be Used in Decomposing Systems into Modules, Communications of the ACM, 15, 1972



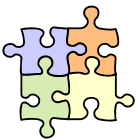
- Obiettivi e argomenti

□ Obiettivi

- introdurre le tattiche architettonali come guida per il raggiungimento di attributi di qualità
- illustrare alcune tattiche architettonali

□ Argomenti

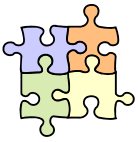
- *tattiche architettonali*
- *tattiche per le prestazioni*
- *tattiche per la modificabilità*
- tattiche per la disponibilità
- tattiche per la sicurezza
- discussione



* Tattiche per la disponibilità

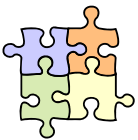
□ La *disponibilità* è interessata ai fallimenti di un sistema e al ripristino da fallimenti

- un *fallimento* si verifica quando il sistema non eroga più un servizio in modo consistente con la sua specifica
 - i fallimenti sono osservabili esternamente dagli utenti del sistema
- un fallimento è causato da un *guasto* (o una loro combinazione)
 - i guasti sono problematiche interne al sistema – ad es., la rottura di un disco o un nodo, o un errore di programmazione
- anche ogni guasto ha la potenzialità di causare un fallimento, non sempre un guasto produce un fallimento
 - un sistema può essere *tollerante* a un certo guasto (o una loro combinazione) se il verificarsi di quel guasto non provoca un fallimento del sistema
- un altro aspetto rilevante è il *ripristino* da fallimenti



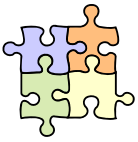
Tattiche per la disponibilità

- In questa trattazione consideriamo *tattiche per la disponibilità* che hanno l'obiettivo di
 - impedire ai guasti di provocare fallimenti
 - o, comunque, di limitare l'effetto del fallimento e rendere possibile il ripristino



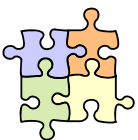
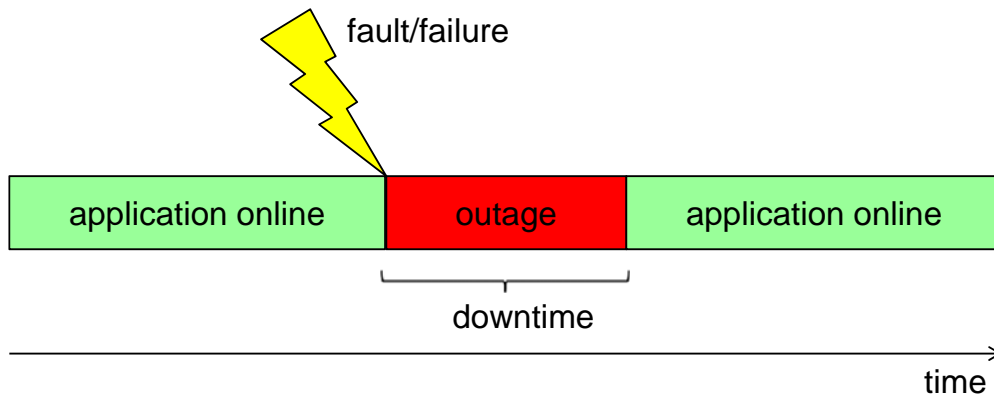
- Che cosa è la disponibilità

- La *disponibilità* può essere definita come
 - la capacità di un sistema di essere completamente o parzialmente funzionante, come e quando richiesto
 - anche a fronte di guasti di componenti del sistema
 - in modo che eventuali guasti non comportino un fallimento totale dell'intero sistema
 - oppure che comportino un fallimento dal quale è possibile un recupero – preferibilmente entro tempi concordati
 - in alcuni casi, può essere accettabile (per un certo periodo di tempo) un funzionamento “parziale” del sistema



Fallimenti

- Un **fallimento** si verifica quando il sistema non eroga più un servizio in modo consistente con la sua specifica
 - si dice che c'è un **interruzione (outage)** di servizio
 - il periodo di tempo in cui il sistema è fuori servizio per un outage è chiamato un **downtime**

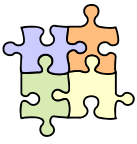


Misura della disponibilità

- La **disponibilità** è definita come un rapporto percentuale

$$\text{disponibilità} = \frac{\text{periodo di tempo in cui il sistema è operativo}}{\text{periodo di riferimento}}$$

Uptime (%)	Downtime (%)	Downtime per year	Downtime per week
98%	2%	7.3 days	3:22 hours
99%	1%	3.65 days	1:41 hours
99.8%	0.2%	17:30 hours	20:10 minutes
99.9%	0.1%	8:45 hours	10:05 minutes
99.99%	0.01%	52.5 minutes	1 minute
99.999%	0.001%	5.25 minutes	6 seconds
99.9999%	0.0001%	31.5 seconds	0.6 seconds

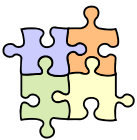


Misura della disponibilità

- La **disponibilità** di un componente può essere calcolata come

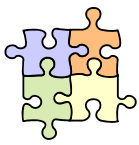
$$\text{disponibilità} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

- **MTBF = Mean Time Between Failures** = inverso del numero di guasti per unità di tempo
 - una proprietà del componente
 - **MTTR = Mean/Maximum Time To Repair or Resolve**
 - tempo medio/massimo impiegato dal fornitore per riparare un guasto al componente dal momento in cui gli viene notificato
- Intuitivamente, è possibile agire sulla disponibilità intervenendo su MTBF e/o su MTTR



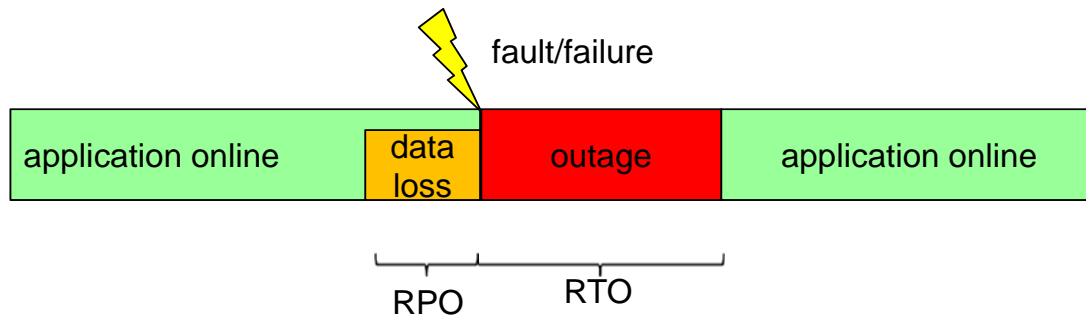
Osservazioni

- In pratica, la caratterizzazione della disponibilità di un sistema è molto più complessa – ad esempio, infatti
 - il sistema può offrire molti servizi, e questi servizi possono fallire in modo indipendente tra loro
 - servizi diversi possono avere requisiti di disponibilità differenti
 - ogni singolo servizio può avere requisiti di disponibilità diversi in momenti (ad es., periodi della giornata) differenti
 - alcuni servizi possono fallire in modo parziale – ovvero, in caso di guasti, può essere accettabile erogare il servizio in modalità “ridotta”
 - il sistema certamente deve prevedere dei periodi di fuori servizio per consentire la manutenzione ordinaria del sistema – chiamati **downtime pianificati**
 - ...



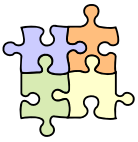
Osservazioni

- Un fallimento può comportare, oltre alla perdita temporanea di funzionalità, anche la perdita di dati “all’indietro nel tempo”
 - ad es., tutte le modifiche fatte dopo l’ultimo backup dei dati
 - gli obiettivi di disponibilità per un sistema possono comprendere allora
 - un tempo massimo (in avanti) per il ripristino delle funzionalità del sistema – *Recovery Time Objective (RTO)*
 - un tempo massimo (all’indietro) di perdita di dati – *Recovery Point Objective (RPO)*



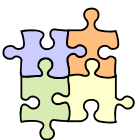
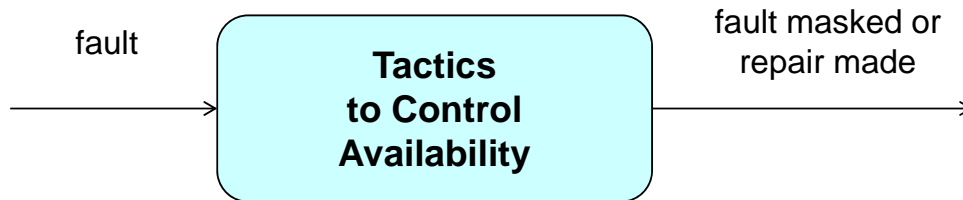
Osservazioni - valutazione economica

- Per *business continuity (BC)* si intende
 - la capacità di un’organizzazione di continuare ad esercitare il proprio business a fronte di guasti o eventi catastrofici
 - la *disponibilità tecnologica* è solo una componente della BC
- La quantità di disponibilità richiesta per un sistema può essere determinata sulla base di una valutazione economica
 - valuta il costo delle interruzioni di servizio
 - a secondo del sistema, un’interruzione di servizio può avere sia conseguenze dirette – ad es., perdita di funzionalità di business, dati, tempo, danni a cose o persone, ... – che indirette – ad es., perdita di clienti, perdita di reputazione, multe, ...
 - sulla base di questa informazione, determina quanto sei disposto a spendere per proteggere il sistema da queste interruzioni di servizio



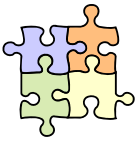
- Tattiche per la disponibilità

- In questa trattazione consideriamo tattiche per la disponibilità per impedire ai guasti di provocare fallimenti, o per limitare l'effetto dei guasti e rendere possibile il ripristino



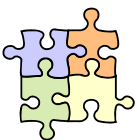
Osservazioni

- Molte tattiche per la disponibilità hanno l'obiettivo di eliminare punti di fallimento singoli
 - per questo, di solito comprendono qualche forma di **ridondanza** – ovvero, la presenza di più “copie” di uno stesso componente
 - il componente in questione può essere un elemento hardware – ad es., un nodo di calcolo, un tratto di rete, una scheda di rete, un elemento di storage o un alimentatore – oppure un componente software (funzionalità oppure dati)
 - ovviamente, è possibile “copiare” più componenti
 - in caso di componenti software, è possibile che le diverse “copie” siano anche diversificate – ad es., funzionalità implementate con algoritmi diversi, oppure dati rappresentati in forme differenti
 - in presenza di copie multiple degli stessi dati, è necessario usare anche dei meccanismi di replicazione – per mantenere allineate queste diverse copie



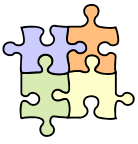
Osservazioni

- Molte delle tattiche per la disponibilità che saranno discusse nel seguito sono presenti in ambienti di esecuzione standard – come sistemi operativi, application server e database server
 - è importante comprendere tali tattiche per considerare esplicitamente (e talvolta individualmente) il loro effetto nella progettazione o nell'analisi di un sistema
- Infine, è bene osservare che molte delle tattiche che saranno discusse nel seguito fanno riferimento principalmente alla disponibilità di nodi (hardware) che implementano servizi
 - e non sempre possono essere applicate direttamente per aumentare la disponibilità dei dati, delle reti, dello storage



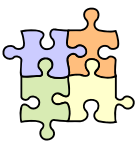
Gruppo di protezione

- Preliminarmente, definiamo un *gruppo di protezione* per un servizio S come un insieme di nodi di elaborazione (server) dedicati all'erogazione del servizio S
 - in un gruppo di protezione, in un dato momento
 - uno o più nodi sono *attivi* – ovvero, sono utilizzati per erogare effettivamente il servizio ai client del servizio
 - possono essere anche presenti altri nodi, che sono delle *riserve* ridondanti
 - il caso più semplice di gruppo di protezione è la ridondanza 1+1
 - un nodo attivo e un nodo di riserva
 - attenzione, una riserva per un servizio S potrebbe essere
 - completamente inattiva, oppure
 - impegnata nei confronti del servizio S – ma non utilizzata direttamente dai client del servizio – oppure
 - impegnata nell'erogazione di altri servizi



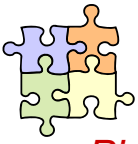
Ragionamento

- Progettazione per la disponibilità
 - viene identificato un albero dei guasti che possono verificarsi
 - viene poi considerato ciascun ramo dell'albero (un guasto o combinazione di guasti) e, anche sulla base della gravità e dell'impatto sui servizi, viene identificata un'opportuna strategia per la gestione di tale guasto
 - attenzione, bisogna fare anche delle considerazioni complessive sulla disponibilità
- Gli approcci per la disponibilità sono solitamente rivolti a
 - prevenire guasti
 - rilevare guasti
 - gestire il ripristino da guasti



Categorie di tattiche per la disponibilità

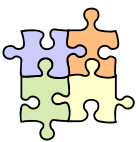
- Categorie principali di tattiche per la disponibilità
 - *fault recovery*
 - tattiche che hanno a che fare con il ripristino del sistema a fronte di guasti
 - ad es., clustering e failover
 - *fault detection*
 - tattiche che hanno lo scopo di rilevare guasti
 - ad es., heartbeat
 - sono propedeutiche alle tattiche per il ripristino automatico – per stabilire se e quando questo debba aver luogo
 - *fault prevention*
 - tattiche che hanno lo scopo di prevenire guasti
 - ad es., riavviare periodicamente un servizio



- Fault detection (1)

▣ *Ping/echo*

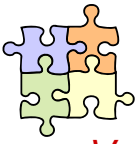
- una coppia di nodi/componenti/processi si scambia in modo asincrono delle coppie di messaggi richiesta/risposta (ping/echo) – per determinare la raggiungibilità e il tempo di roundtrip del canale di comunicazione tra i due elementi
- l'assenza di una risposta indica un problema nell'altro elemento – oppure nella rete di comunicazione
- questa tattica può essere usata da un gruppo di elementi che appartengono ad uno stesso gruppo di protezione, che sono dunque mutuamente responsabili per un certo compito o servizio
 - ad es., per capire se e quando un nodo di riserva debba sostituire un nodo attivo



Fault detection (2)

▣ *System monitor*

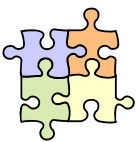
- questa tattica prevede il monitoraggio dello stato di salute del sistema – come la presenza di processi bloccati o fuori controllo
- ad es., un processo/componente emette periodicamente un messaggio *heartbeat* – mentre un altro processo/componente lo ascolta
 - l'assenza di un heartbeat indica un problema nel primo componente (o nella rete di comunicazione)
- ad es., un *watchdog* è basato sull'uso di un contatore o un timer (hardware) che dovrebbe essere periodicamente resettato dal processo (software) monitorato
 - un valore “troppo alto” per questo contatore indica un problema nel processo monitorato



Fault detection (3)

□ Voting

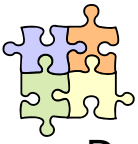
- ci sono più componenti che ricevono gli stessi eventi e inviano le loro risposte ad un componente che gestisce la votazione – questo componente è in grado di rilevare inconsistenze, che indicano guasti
- criteri per lo scrutinio possono essere, ad es., “maggioranza” o “componente preferito”
- poiché tutti i componenti ricevono e elaborano gli stessi eventi, lo stato di questi componenti è continuamente sincronizzato
- nell’hardware, una realizzazione comune di questa tattica è la *triple modular redundancy* (TMR)
 - ad esempio, se un componente ha error rate (inverso dell’MTBF) 10^{-3} , allora l’applicazione di TMR porta ad un error rate pari a circa $2 \cdot 10^{-6}$, ovvero circa 3 ordini di grandezza migliore



Fault detection (4)

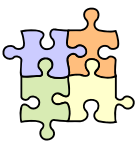
□ Voting (segue)

- nel contesto del software, i componenti sono processi (ad es., in esecuzione su processori ridondanti), talvolta basati su algoritmi diversi (questo consente di controllare eventuali guasti nel software), eventualmente sviluppati da team diversi
 - mentre la ridondanza di elementi uguali tra loro consente di controllare la disponibilità dell’hardware, il controllo della disponibilità del software spesso richiede anche meccanismi di diversificazione



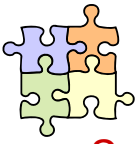
- Fault recovery

- Due categorie di tattiche per il ripristino da guasti
 - tattiche che preparano ed eseguono il ripristino
 - tattiche relative alla reintroduzione del componente che si è guastato
 - la riparazione può essere automatica o richiedere un intervento manuale
 - la riparazione automatica richiede evidentemente anche l'applicazione di tattiche per la fault detection



Fault recovery (1)

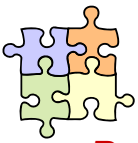
- *Active redundancy (hot spare)*
 - si riferisce ad una configurazione in cui tutti i nodi (attivi o riserve) in un gruppo di protezione ricevono ed elaborano in parallelo gli stessi identici eventi – pertanto sono normalmente tutti nello stesso stato
 - se si verifica un guasto in un nodo attivo, uno dei nodi di riserva lo può sostituire immediatamente
 - *il downtime del sistema può essere nell'ordine dei millisecondi*
 - il caso più semplice è la ridondanza 1+1
 - poiché tutti i nodi ricevono gli stessi eventi, la loro sincronizzazione è automatica – è però opportuno usare dei protocolli di comunicazione affidabili tra di essi
 - la ridondanza può essere anche nel canale di comunicazione (rete) e nelle unità disco (condivise)



Fault recovery (2)

▣ *Spare (cold spare)*

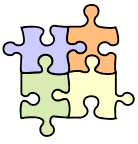
- si riferisce ad una configurazione in cui i nodi riserva di un gruppo di protezione sono tenuti fuori servizio fino a quando non è necessaria la sostituzione di un nodo attivo
 - i nodi di riserva sono configurati per poter sostituire altri nodi che potrebbero fallire
- quando si verifica un guasto in un nodo attivo, il nodo di riserva viene avviato, eventualmente configurato, e il suo stato aggiornato (vedi ad esempio *Rollback*) – solo allora può sostituire il nodo che è fallito
 - *il downtime del sistema può essere nell'ordine dei minuti*



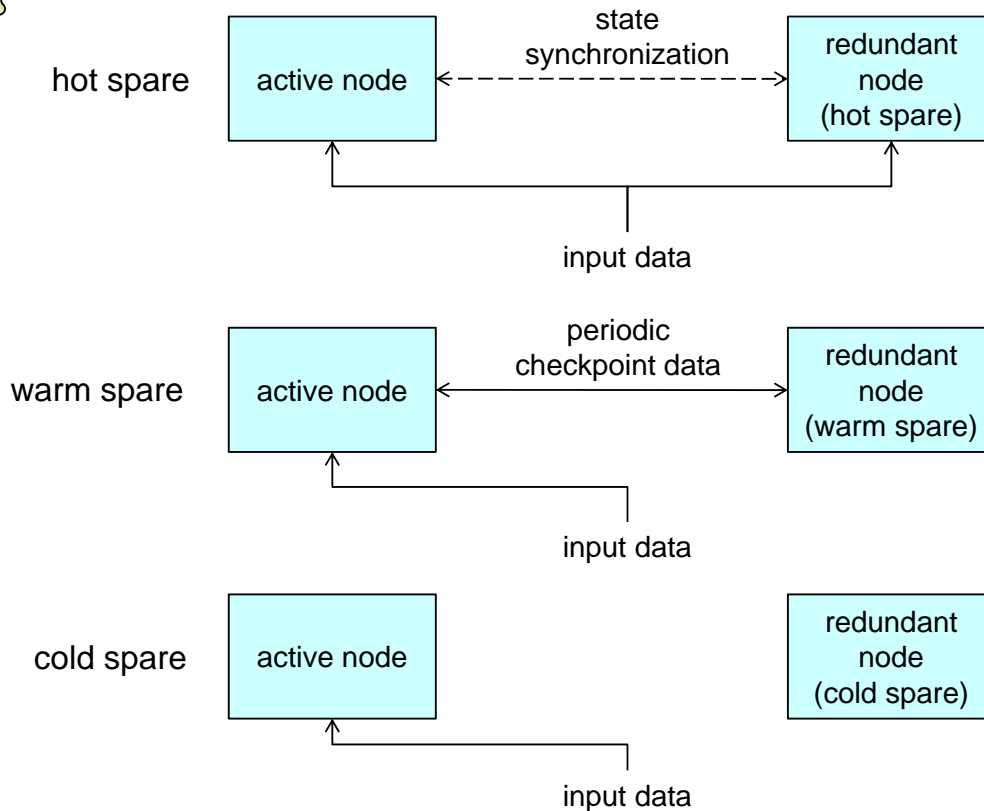
Fault recovery (3)

▣ *Passive redundancy (warm spare)*

- in questa configurazione, solo i nodi attivi del gruppo di protezione ricevono ed elaborano gli eventi di input – inoltre i nodi attivi notificano periodicamente i nodi di riserva dei cambiamenti di stato che si sono verificati
- se si verifica un guasto in un nodo attivo, uno dei nodi di riserva lo può sostituire – ma solo dopo che il suo stato è stato opportunamente aggiornato
- la disponibilità può essere aumentata forzando periodicamente lo scambio attivo-riserva – ad esempio, giornalmente, dopo ogni ora, oppure anche dopo ciascuna transazione
- i nodi attivi hanno anche la responsabilità di controllare la sincronizzazione – anche sulla base di un meccanismo di broadcast affidabile
- è un compromesso (affidabilità/complessità) tra le due tattiche precedenti – *il downtime può essere nell'ordine dei secondi*



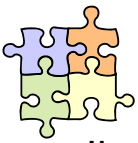
Hot spare, warm spare, cold spare



27

Tattiche architetturali

Luca Cabibbo - ASw



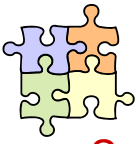
Clustering

- Il **clustering** è una configurazione per la disponibilità di uso comune, e può assumere anche una forma diversa da quelle mostrate
 - le tecniche di clustering prevedono di usare più nodi per erogare attivamente un certo servizio
 - è presente anche un elemento di bilanciamento del carico, che ha la responsabilità di assegnare le diverse richieste al servizio (da parte di molteplici client) ai nodi del cluster
 - possono essere necessari meccanismi per sincronizzare lo stato tra i nodi di un cluster
 - l'obiettivo di queste tecniche è sostenere sia disponibilità che scalabilità
 - considereremo il clustering nel contesto degli stili architetturali per sistemi distribuiti

28

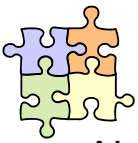
Tattiche architetturali

Luca Cabibbo - ASw



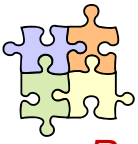
Fault recovery (4)

- *Software upgrade*
 - questa tattica ha l'obiettivo di effettuare aggiornamenti al codice eseguibile – eventualmente senza nessuna interruzione di servizio
 - ad esempio, l'aggiornamento viene effettuato dapprima su un nodo di riserva, che poi (dopo una risincronizzazione dello stato) sostituisce il corrispondente nodo attivo
 - alcuni ambienti di esecuzione offrono delle modalità di aggiornamento online più sofisticate



Fault recovery (5)

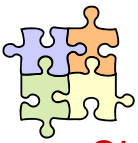
- Alcune tattiche di ripristino dai guasti sono relativi alla reintroduzione di componenti che si sono guastati – che vengono reintrodotti dopo essere stati “corretti”
- *State resynchronization*
 - le tattiche di ridondanza attiva e passiva richiedono che i vari nodi attivi e di riserva siano tutti continuamente nello stesso stato
 - pertanto, prima di reintrodurre in servizio un nodo, è necessario verificare la correttezza del suo stato oppure ripristinarlo



Fault recovery (6)

▣ *Rollback*

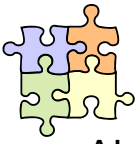
- un checkpoint è la registrazione di uno stato consistente che avviene periodicamente oppure in risposta ad eventi specifici
- un log delle transazioni è la registrazione di tutti i cambiamenti di stato avvenuti
- questa tattica è relativa ad un meccanismo di ripristino dello stato basata su checkpoint e log delle transazioni



Fault recovery (7)

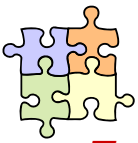
▣ *Shadow*

- un nodo in precedenza fallito (ed è stato reintrodotta) viene temporaneamente eseguito in “modalità ombra” (come riserva) per verificare la correttezza del suo comportamento, prima di essere reintrodotta come attivo



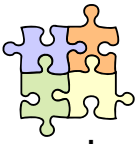
- Fault prevention (1)

- Alcune tattiche per la prevenzione dei guasti
- *Removal from service*
 - questa tattica rimuove temporaneamente un componente del sistema dalle operazioni per fargli svolgere attività che possono mitigare possibili fallimenti di sistema
 - ad esempio, effettuare un reboot per fare in modo che sfridi o perdite nell'allocazione di memoria causino dei fallimenti
- *Process monitor*
 - tattica relativa al monitoraggio dello stato di salute di processi – quando viene rilevato un guasto in un processo, questo può essere cancellato e poi ricreato
 - è ad un livello gerarchico più basso di *System monitor*



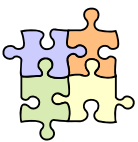
Fault prevention (2)

- *Transactions*
 - i servizi ad alta disponibilità sono solitamente basati su una semantica transazionale – una transazione è una sequenza di passi elementari che viene complessivamente considerata un'operazione atomica – da svolgere oppure annullare completamente
 - consente di prevenire inconsistenze nello stato del sistema, nel caso di fallimento di uno dei passi elementari
 - ad esempio, 2PC nel caso di transazioni distribuite



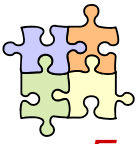
- Exceptions (1)

- Le **eccezioni** sono un meccanismo dei linguaggi di programmazione per gestire situazioni “eccezionali” rispetto al flusso di esecuzione normale del programma – dunque particolarmente adatto alla gestione di guasti
 - è dunque in genere opportuno applicare le eccezioni nello sviluppo di software con requisiti di disponibilità
 - tre tattiche per l’applicazione delle eccezioni
 - ciascuna contestualizza l’applicazione delle eccezioni ad una delle tre categorie di tattiche introdotte in precedenza



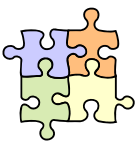
Exceptions (2)

- **Exception detection** – per fault detection
 - il rilevamento di eccezioni si riferisce al rilevamento di condizioni del sistema che, appunto, devono alterare il normale flusso di esecuzione
 - ad esempio, eccezioni di sistema (es., divisione per zero, errore nell’accesso alla memoria), oppure eccezioni legate alla verifica di codici di controllo nei dati scambiati tra componenti o della correttezza delle risposte calcolate da un algoritmo

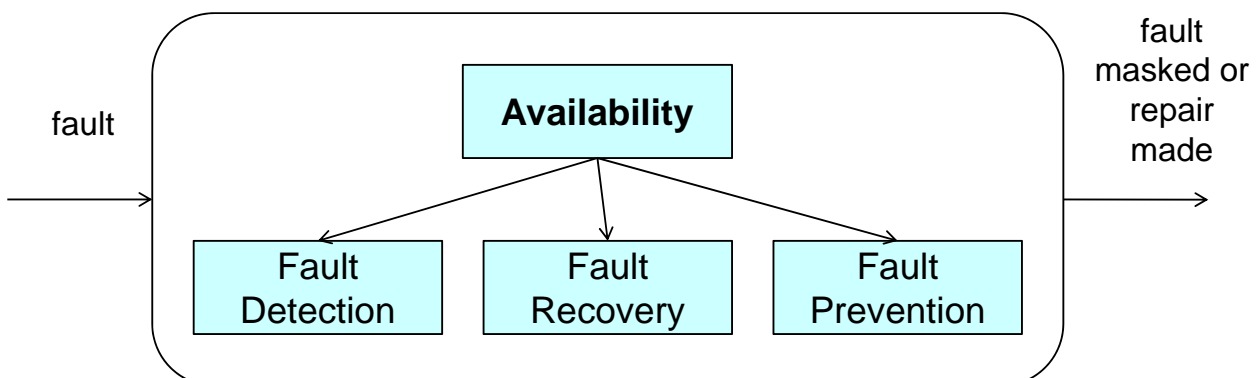


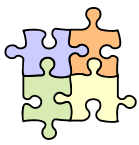
Exceptions (3)

- **Exception handling** – per fault recovery
 - la gestione di eccezioni può essere usata per il ripristino da guasti – notificati appunto come eccezioni
 - il tipo, l'origine e la causa dell'eccezione possono essere usate per selezionare un'opportuna modalità di ripristino
- **Exception prevention** – per fault prevention
 - questa tattica ha lo scopo di prevenire il verificarsi di eccezioni
 - ad esempio, usare un linguaggio che prevede un insieme di controlli statici forti – oppure, l'uso di tipi che controllano tutti gli accessi tramite puntatori, per evitare riferimenti appesi

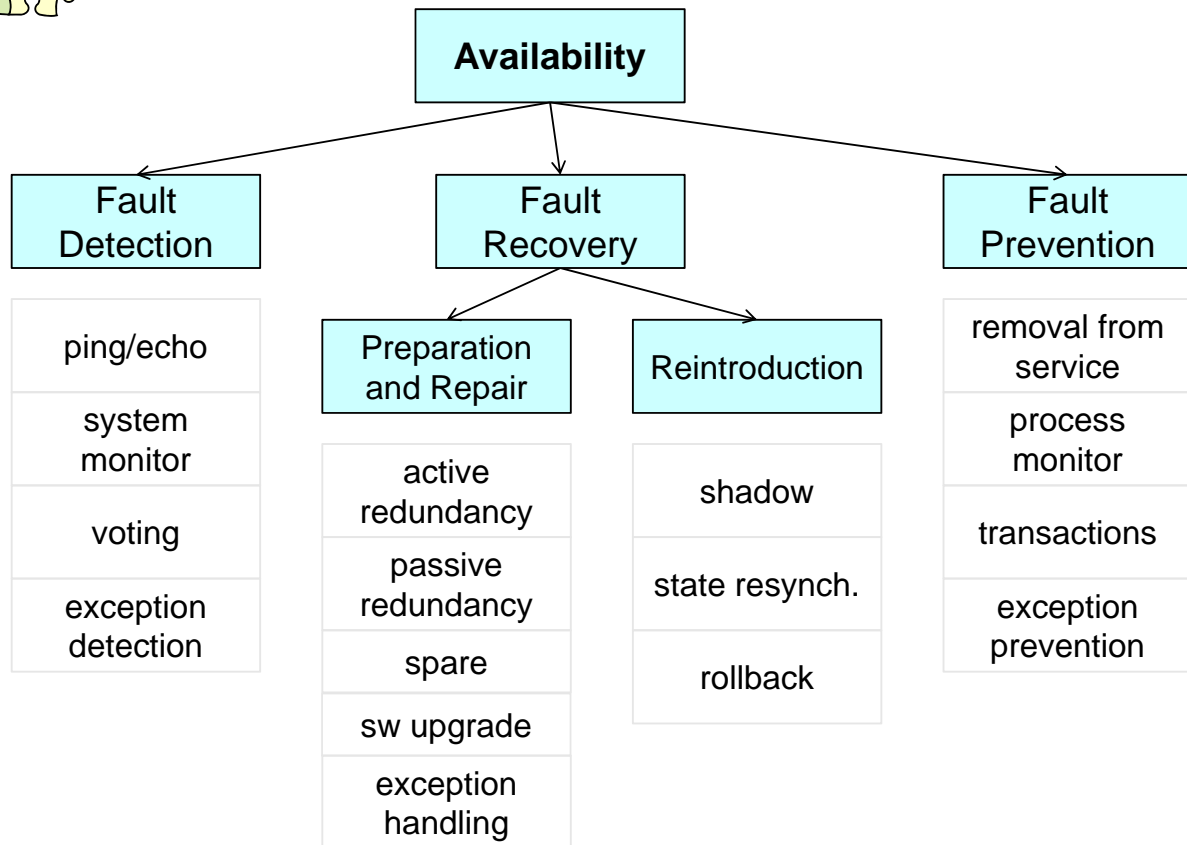


- Tattiche per la disponibilità - sintesi





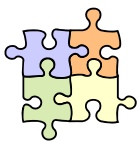
Tattiche per la disponibilità - sintesi



39

Tattiche architetturali

Luca Cabibbo - ASw



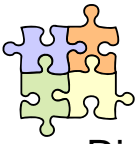
Tattiche per la disponibilità - discussione

- ▣ Riguardo alle tattiche per la disponibilità
 - l'applicazione di una tattica può richiedere anche l'applicazione di altre tattiche – in altre categorie
 - ad esempio, le tattiche di ripristino da guasti (hot spare, warm spare e cold spare) vanno sicuramente applicate insieme a tattiche di rilevamento di guasti – per consentire un ripristino automatico
 - l'applicazione di una tattica non impedisce l'applicazione di tattiche – anche nella stessa categoria
 - ad esempio, la realizzazione di un servizio altamente disponibile può richiedere sia una ridondanza hot spare (nello stesso sito) che una ridondanza warm spare (in un sito remoto di disaster recovery)

40

Tattiche architetturali

Luca Cabibbo - ASw



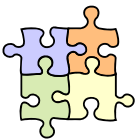
Tattiche per la disponibilità - discussione

- ▣ Riguardo alle tattiche per la disponibilità
 - l'applicazione di una tattica per la disponibilità può essere efficace a fronte di un certo tipo di guasto – ma non lo è necessariamente con tutti i possibili guasti (allo stesso modo)
 - ad es., la replicazione di un server in un rack consente di gestire la rottura del server – ma non scenari come un incendio, un allagamento o un'interruzione di energia elettrica
 - ad es., una certa configurazione potrebbe essere in grado di gestire la rottura di un server in pochi millisecondi – ma la rottura di un'unità di storage in diverse ore
 - molte configurazioni richiedono considerazioni speciali
 - ad es., se un processo è replicato ed è in esecuzione su due nodi “virtuali” – allora è bene che i due nodi “virtuali” siano allocati su due nodi “fisici” distinti

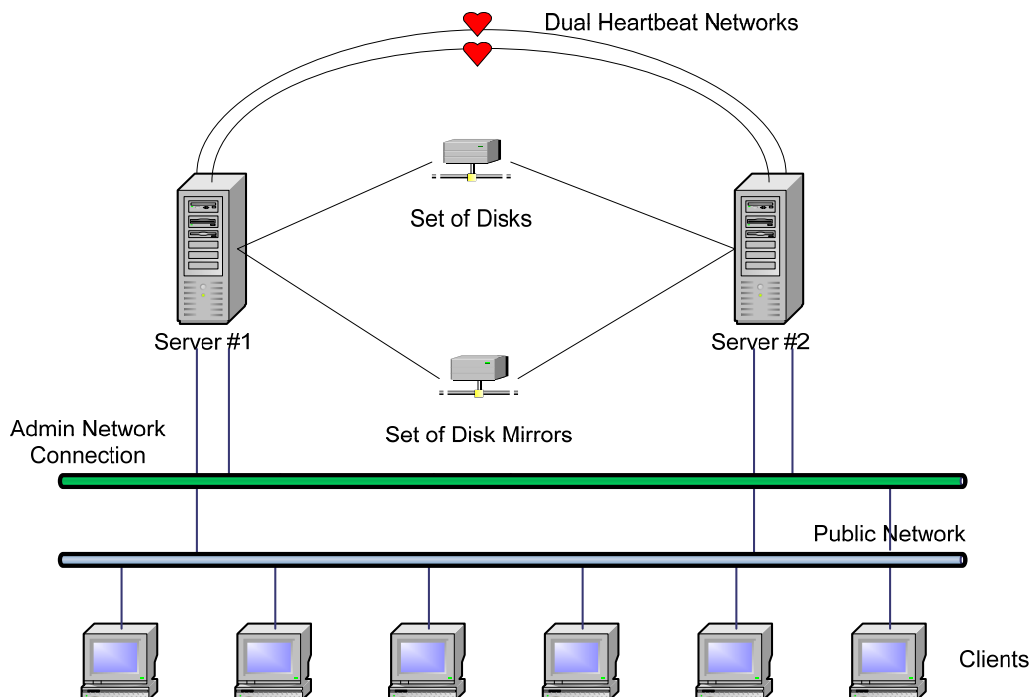
41

Tattiche architetturali

Luca Cabibbo – ASw



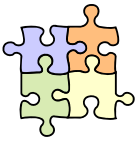
Esempio - un semplice cluster



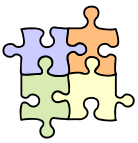
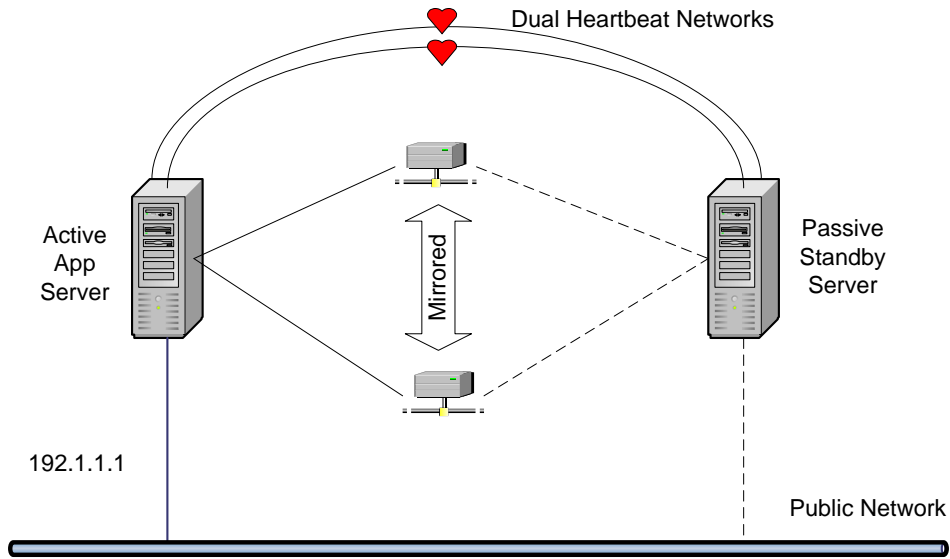
42

Tattiche architetturali

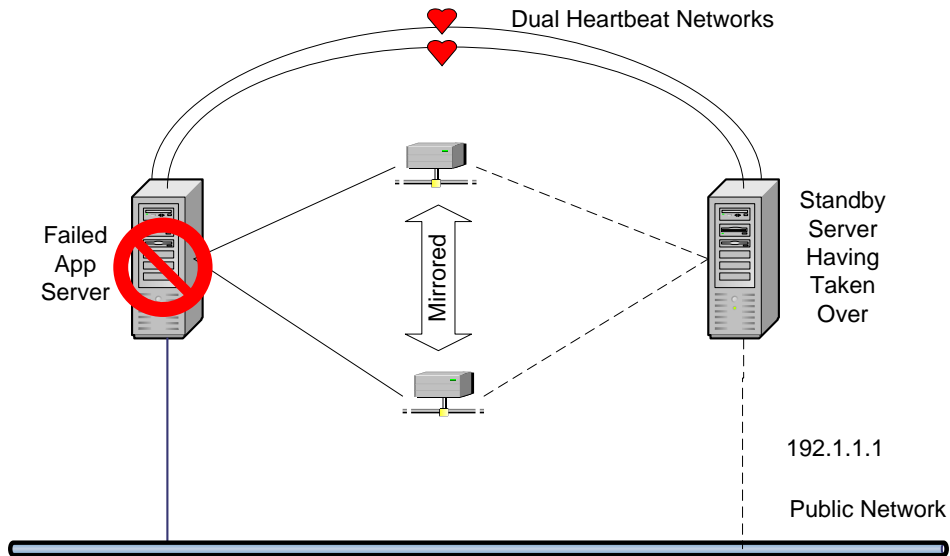
Luca Cabibbo – ASw

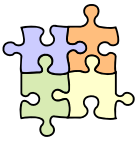


Config. active-passive - prima di un failover

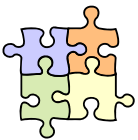
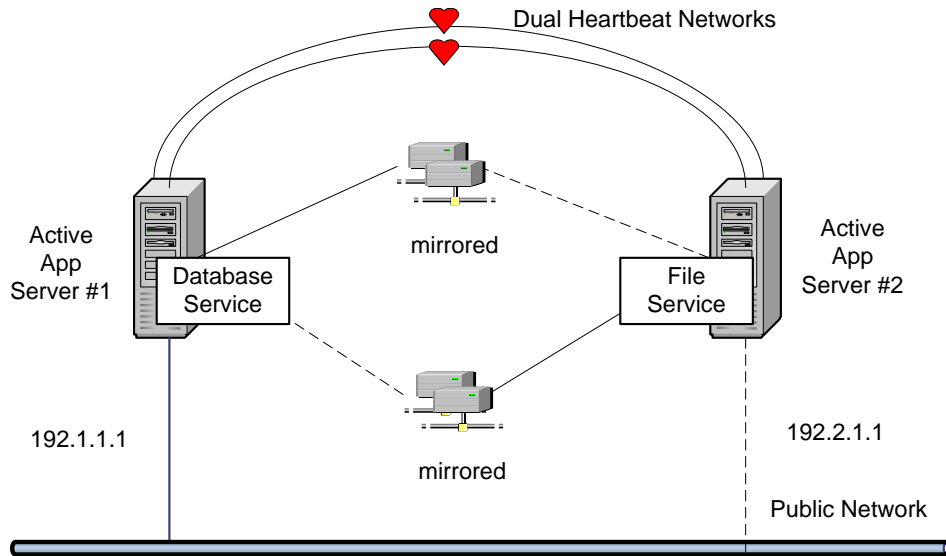


Config. active-passive - dopo un failover

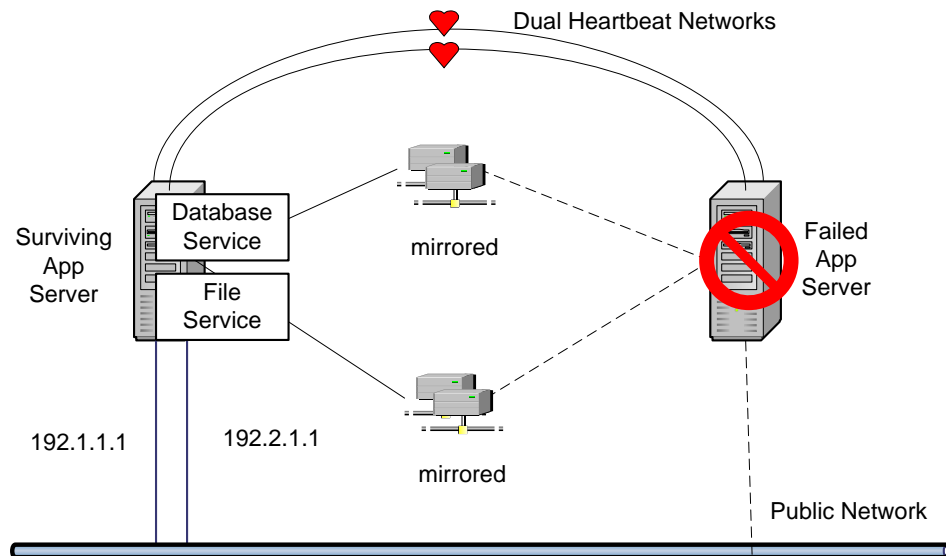


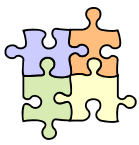


Config. active-active - prima di un failover



Config. active-active - dopo un failover





* Tattiche per la sicurezza

- La **sicurezza** è relativa alla capacità del sistema di resistere ad usi non autorizzati – e nel tempo stesso fornire servizi ai suoi utenti legittimi
- In questa trattazione consideriamo alcune **tattiche per la sicurezza** che hanno l'obiettivo di controllare alcuni dei seguenti aspetti
 - **confidenzialità/segretezza** – occultamento di informazioni o risorse sensibili (delicate)
 - **integrità** – fidatezza nelle informazioni o nelle risorse
 - integrità dei dati e dei programmi
 - integrità dell'origine – ovvero, dell'identità partecipanti alle transazioni
 - **disponibilità** – la capacità di usare le informazioni o le risorse desiderate
 - **accountability** – sapere chi ha avuto accesso ad informazioni o risorse

47

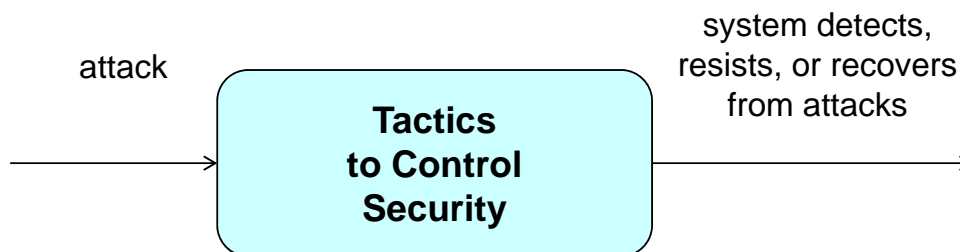
Tattiche architetturali

Luca Cabibbo – ASw



Tattiche per la sicurezza

- Obiettivo delle tattiche per la sicurezza che sono descritte in questa trattazione

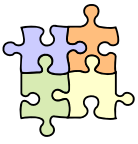


- un **attacco** è un tentativo di rompere la sicurezza di un sistema

48

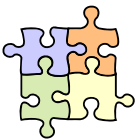
Tattiche architetturali

Luca Cabibbo – ASw



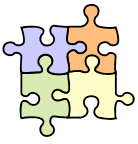
Categorie di tattiche per la sicurezza

- Tre categorie principali di tattiche per la sicurezza – solitamente da usare in combinazione
 - resistere agli attacchi – *resisting attacks*
 - simile all'uso di una serratura – obiettivo è tenere fuori i “cattivi”
 - rilevare attacchi – *detecting attacks*
 - simile all'uso di un allarme – i “cattivi” possono entrare, ma non possono fare danni
 - ripristino da attacchi – *recovery from attacks*
 - simile all'uso di un'assicurazione – i “cattivi” possono entrare e fare danni, ma i danni possono essere riparati



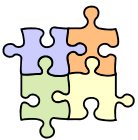
- Resisting attacks (1)

- Tattiche per resistere agli attacchi
 - *Authenticate users*
 - l'autenticazione è assicurarsi che l'utente o sistema remoto è chi dichiara di essere
 - ad es., password, certificati, ...
 - *Authorize users*
 - le autorizzazioni sono usate per verificare che un utente autenticato abbia degli opportuni diritti di accesso nei confronti delle risorse (dati o servizi) che intende utilizzare
 - ad es., acl (access control lists), autorizzazioni per utenti, gruppi di utenti o ruolo, ...



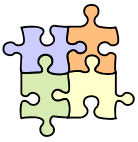
Resisting attacks (2)

- *Maintain data confidentiality*
 - protezione dei dati da un accesso non autorizzato – solitamente basata su qualche forma di cifratura
 - ad es., cifratura simmetrica o asimmetrica, ...
 - attenzione al costo temporale della cifratura
 - è in contrapposizione con *Reduce computational overhead*
- *Maintain integrity*
 - i dati devono essere consegnati come sono
 - solitamente basato sull'uso di informazioni di controllo ridondanti, certificati, cifratura, ...



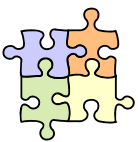
Resisting attacks (3)

- *Limit exposure*
 - i servizi sono essere distribuiti su più host, in modo tale che un attacco possa colpire i servizi solo in modo parziale o in numero limitato
 - infatti gli attacchi sono solitamente rivolti verso singole debolezze del sistema
- *Limit access*
 - un firewall può restringere l'accesso sulla base della sorgente del messaggio o la porta di destinazione
 - consente di limitare l'accesso nei confronti di client non conosciuti
 - per limitare l'accesso da client conosciuti (ad es., dal proprio web server pubblico) è possibile usare configurazioni come, ad es., una zona demilitarizzata (DMZ)



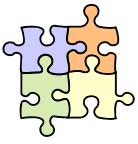
- Detecting attacks

- Il rilevamento di attacchi viene solitamente fatto mediante un sistema di rilevamento delle intrusioni
 - ad esempio, che confronta il traffico dei messaggi con pattern di dati storici di attacchi conosciuti

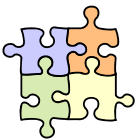
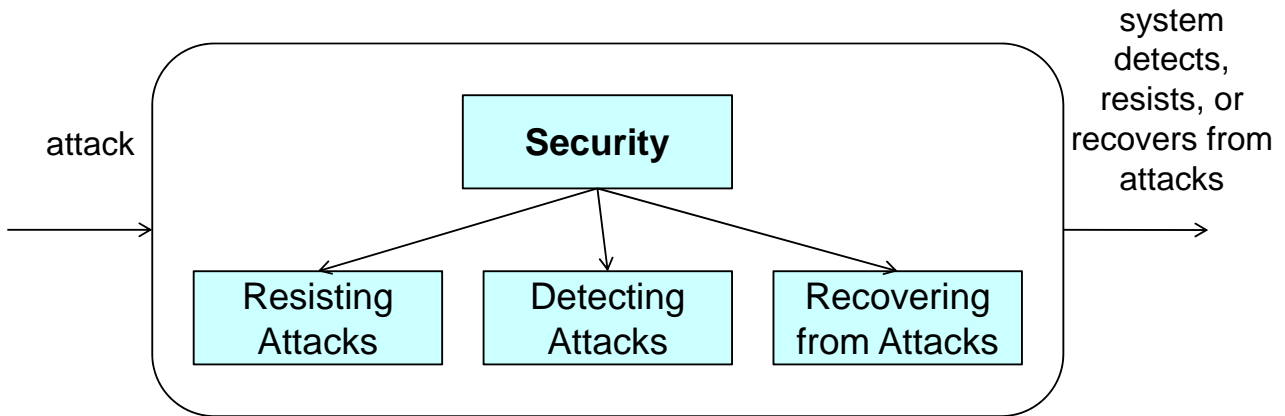


- Recovering from attacks

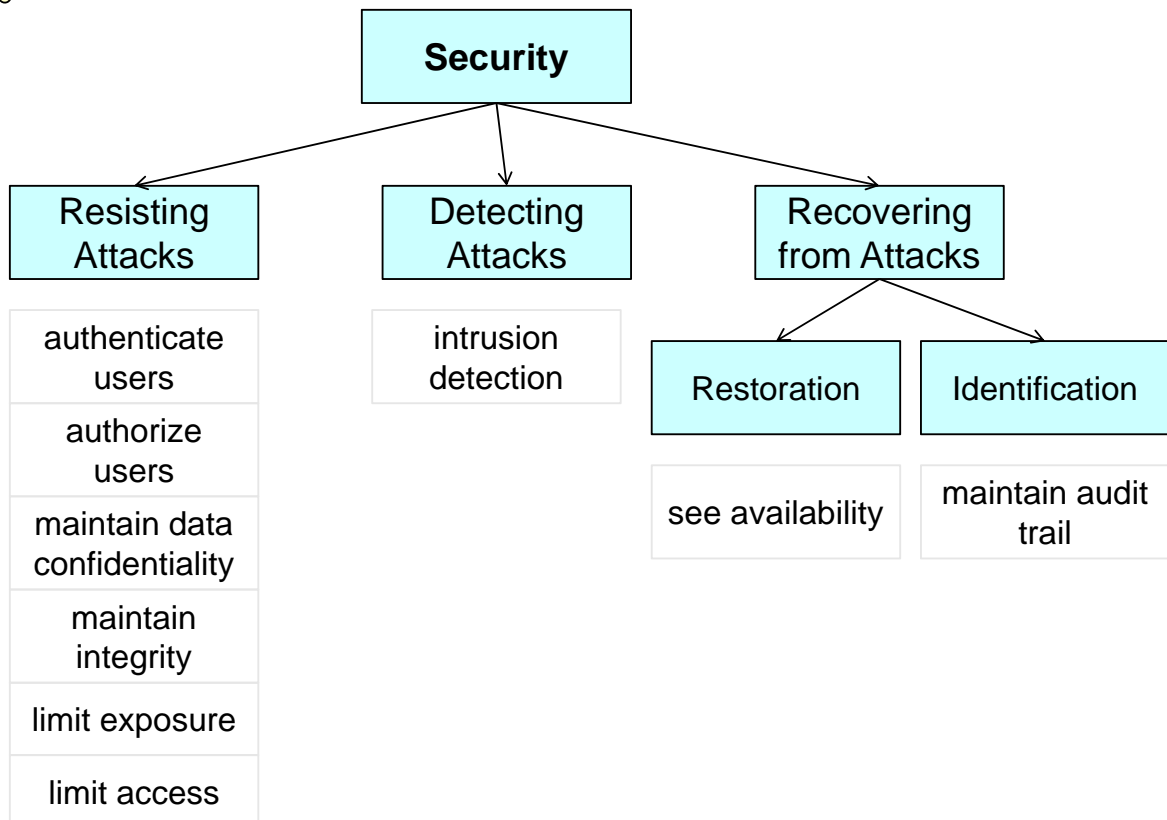
- Due categorie di tattiche per il ripristino da attacchi
 - tattiche che hanno lo scopo di ripristinare lo stato del sistema
 - vedi le tattiche corrispondenti per la disponibilità
 - tattiche che hanno lo scopo di identificare l'origine dell'attacco – a fini punitivi
 - solitamente sulla base di informazioni di auditing, ovvero la registrazione degli accessi al sistema e degli utenti che li hanno effettuati

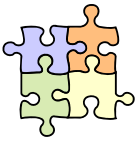


- Tattiche per la sicurezza - sintesi



Tattiche per la sicurezza - sintesi





* Discussione

- Sono state presentate alcune tattiche architettoniche – nel contesto del controllo di alcuni attributi di qualità
 - applicare una tattica vuol dire prendere una decisione di progetto per controllare un certo attributo di qualità
 - questo ha impatto sull'architettura – ovvero, sulla scelta degli elementi, delle loro responsabilità, o di come sono messi in relazione – e su come l'architettura supporta le qualità
 - la presentazione è stata qualitativa e informale
 - sono talvolta disponibili modelli che descrivono l'effetto quantitativo dell'applicazione delle tattiche
 - è inoltre possibile modellare queste decisioni di progetto – anche sulla base dell'applicazione di scenari
 - sono stati mostrati alcuni compromessi che occorre valutare nel controllo congiunto di più attributi di qualità
 - esistono altri attributi di qualità ed altre tattiche



Discussione

- La progettazione di un'architettura richiede l'applicazione di una collezione di tattiche, per realizzare una **strategia architettonica**
 - l'approccio (solitamente di compromesso) adottato al fine di raggiungere gli obiettivi complessivi di qualità del sistema
- Un altro approccio fondamentale per la progettazione dell'architettura è basato sugli stili architettonici
 - uno stile architettonico può essere basato sull'applicazione di un certo numero di tattiche
 - nel seguito del corso questa affermazione sarà esemplificata nel contesto dello studio degli stili architettonici