UNIVERSITÀ DI PARMA

Dipartimento di Ingegneria e Architettura
Corso di Laurea Magistrale in Ingegneria Informatica

CONTRACT NET PROTOCOL PROJECT

Professore:
Chiar.mo Prof. AGOSTINO POGGI

Studenti:
Alex Tivoli
Filippo Botti

ANNO ACCADEMICO 2022-2023

Indice

Introduzione

Questo progetto rappresenta un esempio di implementazione del protocollo Contract Net utilizzando il framework ActoDes. Il Contract Net Protocol è un protocollo di condivisione delle attività nei sistemi multi-agente, introdotto nel 1980 da Reid G. Smith. Questo protocollo viene utilizzato per allocare compiti tra agenti autonomi e coinvolge un manager (master) e alcuni fornitori di servizi (workers). Il manager propone l'esecuzione di un compito (task announcement) ai lavoratori che conosce e i lavoratori possono fare proposte (bid) o indicare la loro indisponibilità. Il manager può assegnare l'incarico a uno o più lavoratori.
Il compito delegato dal Manager ai workers è stato il calcolo di numeri appartenenti alla sequenza di fibonacci.
L'elaborato è stato realizzato sfruttando il software ActoDes e ha coinvolto da 1 a 5 workers. Il progetto prevedeva due tipi di esecuzione:

- Un'esecuzione senza memoria, dove non era previsto il salvataggio dei risultati della sequenza di fibonacci calcolata nei task precedenti dai workers
- Un'esecuzione con memoria, dove il salvataggio era previsto e poteva essere sfruttato per diminuire il costo dei task successivi

La seguente relazione analizzerà la soluzione proposta e i risultati ottenuti. È inoltre riportata una guida per l'installazione e l'esecuzione del codice.

Capitolo 1

Soluzione Realizzata

Il progetto consiste in tre classi principali: Initiator, Master e Worker. In figura 1.1 è riportato uno schema riassuntivo dell'esecuzione prevista.

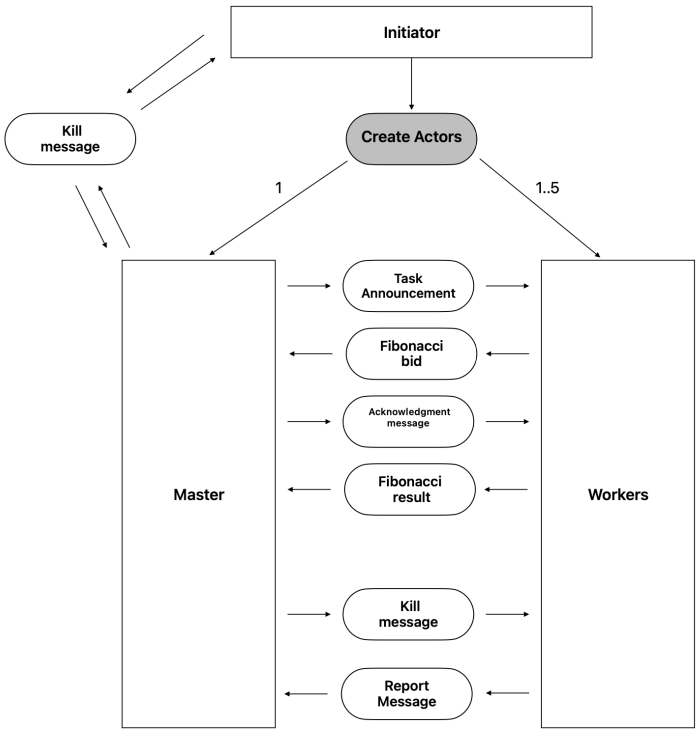

Figura 1.1: Schema riassuntivo dell'esecuzione prevista

1.1 Initiator

Questa classe si occupa di creare un'instanza della classe Master e N instanze (al più cinque) della classe Worker. La classe Initiator ha il compito di far partire e terminare tutta l'applicazione.

1.2 Master

Questa classe definisce il comportamento del manager. Il Master ad ogni iterazione genera un numero random compreso tra 2 e 100 e manda ad ogni workers la richiesta di calcolo della sequenza di Fibonacci del numero generato, aspettandosi come risposta da ogni workers attivo una proposta basata sul costo di calcolo. Il Master sceglie i workers da accettare in base alle proposte ricevute, prediligendo l'offerta minore. Se nessuna offerta è ricevuta, il master genera un nuovo numero su cui calcolare la sequenza di Fibonacci. Il Master inoltre salva ogni proposta accettata con il relativo costo, per generare un report finale. Dopo 50 iterazioni in cui il master ha accettato una o più proposte, esso invia il messaggio di chiusura ad ogni workers e anche all'Initiator, e aspetta di essere terminato.

1.3 Worker

Questa classe rappresenta il comportamento del Worker, che aspetta i messaggi dal master fino a quando riceve un messaggio di terminazione. Quando ciò accade, il Worker termina la propria esecuzione. Il Worker può inviare offerte al master e, dopo un timeout, se è stata accettata l'offerta dal Master, svolgere il compito specifico, ovvero calcolare e restituire al Master il calcolo della sequenza di Fibonacci del numero presente nella richiesta del Master. Prima di eseguire il calcolo effettivo, nel caso lo storage sia abilitato, il Worker calcola il costo valutando se non ha già generato la sequenza fino al numero della richiesta, al fine di abbassare l'offerta e aumentare le probabilità di essere scelto. L'offerta finale è data dalla somma del costo calcolato

più una costante k moltiplicata per 5. Questa costante vuole rappresentare il concetto di *fairness*: ogni volta che il worker disponibile non viene scelto il k diminuisce, andando anche in negativo, al fine di abbassare l'offerta.

1.4 Funzionamento

Vediamo in questa sezione una breve descrizione del funzionamento del sistema proposto. Dopo aver eseguito la classe Initiator e specificato via input il numero di worker e l'abilitazione o meno dello storage, vengono creati gli attori: Master e Workers.
Si procede quindi con il Master che comunica a tutti i worker il numero della sequenza di fibonacci da calcolare, attraverso un **task announcement**. Ogni worker, ricevuto il task announcement, simula la propria disponibilità attraverso un valore random (tra 0 e 1). In entrambi i casi viene inviato al Master un messaggio di tipo **fibonacci bid**, contenente l'offerta proposta. L'offerta è pari a -1 nel caso in cui il worker manifestasse la propria indisponibilità.
Dopo aver ricevuto tutte le offerte il Master seleziona le offerte a lui più convenienti (le meno costose) ed invia un messaggio di tipo **message acknowledgment** ai worker che le hanno presentate.
I worker alla ricezione dell'acknwoledgment settano a true il flag corrispondente all'offerta accettata. Allo scadere del timer impostato durante l'invio dell'offerta (fibonacci bid) i worker controllano il valore del flag e nel caso in cui sia posto a true cominicano il calcolo della sequenza di fibonacci (sfruttando lo storage se abilitato).
Dopo aver calcolato la sequenza di fibonacci la inviano al Master attraverso un messaggio di tipo **fibonacci result**.
Quando il Master riceve il risultato di fibonacci verifica il numero dei task eseguiti e se pari a 50 invia un messaggio di **kill** ai worker. I worker ricevuto il messaggio di kill rispondono al Master con un **report message** contenente lo storico delle offerte accettate e successivamente terminano l'esecuzione.

Il Master ricevuti tutti i report procede alla stesura del report finale e invia all'Initiator un messaggio di **kill**. L'Initiator risponde a sua volta con un messaggio di kill al Master e successivamente termina l'esecuzione del sistema.

Capitolo 2

Risultati

Nel seguente capitolo andiamo ad analizzare le varie esecuzioni. Il progetto infatti consisteva nel valutare vari tipi di esecuzione, variando il numero di workers e abilitando o meno il salvataggio delle sequenze di fibonacci precedenti. Ogni esecuzione comprendeva 50 task accettati dal master. Per ogni esecuzione è disponibile un report testuale che indica i seguenti valori:

- Spesa totale del Master
- Spesa media per task del Master
- Conteggio delle esecuzioni simultanee di due o più workers (ovvero quante volte due o più workers hanno inviato un'offerta uguale)
- Trend dei prezzi accettati dal Master
- Tempo di esecuzione totale
- Trend dei guadagni di ogni Worker
- Guadagno medio di ogni worker
- Guadagno medio totale di ogni worker

I valori significativi sono stati riportati nelle tabelle 2.1 e 2.2. Come previsto le esecuzioni con storage abilitato permettono un notevole risparmio da parte

del Master, indipendentemente dal numero di workers coinvolti nel sistema. Questo è dato dalla natura dello storage: andando a salvare le sequenze passate i costi di esecuzione si abbassano e di conseguenza anche le offerte dei workers, così da permettere al master di accettare proposte meno costose.
Il concetto di offerta nell'elaborato è strettamente collegato al numero di fibonacci da calcolare: infatti, se non presente nello storage, l'offerta è pari al numero richiesto (tralasciando il concetto di fairness spiegato precedentemente). Considerato 5 il minimo valore di un'offerta (nel caso in cui il numero sia già presente nello storage o l'offerta sia molto bassa per il coefficiente di fairness), per evitare offerte a *costo negativo*. Il coefficiente di fairness si dimostra utile nel cercare di mantenere uguale tra tutti i worker il guadagno medio, rimuovendolo potrebbe causare massimi guadagni in un singolo worker, penalizzando gli altri presenti nel sistema.
I tempi di esecuzione sono costanti per ogni esecuzione nel caso in cui la sequenza di fibonacci sia calcolata sfruttando il metodo iterativo, sono invece notevolmente diversi quando fibonacci è calcolato con il metodo ricorsivo: le esecuzioni con storage terminano notevolmente prima rispetto a quelle con storage disabilitato, dato che ogni volta la sequenza viene ricalcolata. Questo lo si nota ancora di più nel caso in cui, con storage abilitato e un solo worker, venga richiesto tra i primi task un numero elevato: in questo modo la computazione viene spostata tutta all'inizio, riducendo a una ricerca all'interno di una hash-map la maggior parte dei successivi task.

Numero Worker	**Spesa totale Master**	**Spesa Media Master**
1 - storage	**359**	**7.18**
2 - storage	503	10.06
3 - storage	646	12.92
4 - storage	667	13.34
5 - storage	745	14.9

Tabella 2.1: Comparazioni tra i valori delle varie esecuzioni con storage abilitato.

Numero Worker	Spesa totale Master	Spesa Media Master
1 - no storage	**2291**	**45.82**
2 - no storage	3030	60.6
3 - no storage	3891	77.82
4 - no storage	3775	75.5
5 - no storage	5920	118.4

Tabella 2.2: Comparazioni tra i valori delle varie esecuzioni con storage disabilitato. Si noti la differenza sostanziale tra le spese del master

2.1 Report

Di seguito vengono riportati i report di ogni esecuzione ed il grafico relativo all'andamento delle spese del Master.

2.1.1 Storage abilitato 5 Workers

Number of workers: 5 **Storage:** true **Execution Time:** 153.317 seconds

- **000000001.12443@192.168.1.3 Mean:** 4.428571 **bids:** [5, 1, 5, 5, 5, 5, 5]
- **000000004.12443@192.168.1.3 Mean:** 7.866667 **bids:** [5, 5, 1, 5, 57, 5, 5, 5, 0, 5, 5, 5, 5, 5, 5]
- **000000002.12443@192.168.1.3 Mean:** 8.068966 **bids:** [40, 18, 36, 5, 5, 5, 5, 5, 5, 5, 9, 5, 5, 5, 11, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
- **000000005.12443@192.168.1.3 Mean:** 9.272727 **bids:** [63, 39, 6, 5, 5, 5, 5, 5, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
- **000000003.12443@192.168.1.3 Mean:** 8.777778 **bids:** [40, 5, 40, 5, 5, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

Master :: Total price: 745 **Average transaction price:** 14.9 **Simultaneous transaction count:** 28

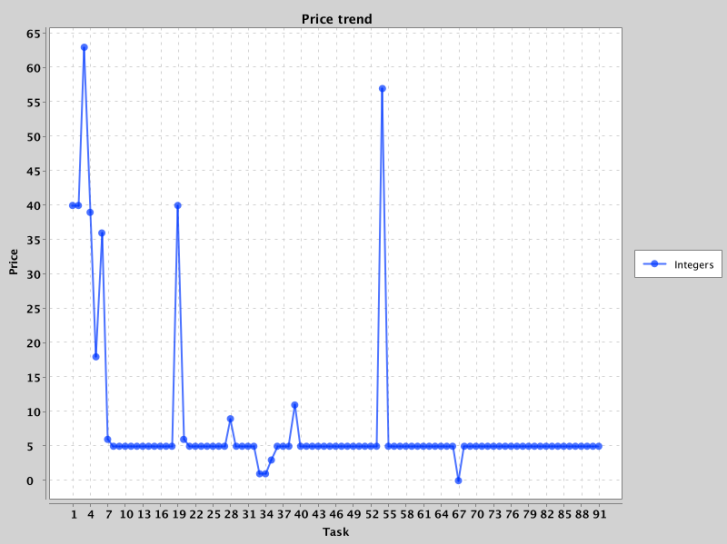

Figura 2.1: Andamento spese del Master con 5 worker con storage abilitato

2.1.2 Storage abilitato 4 Workers

Number of workers: 4 **Storage:** true **Execution Time:** 153.56 seconds

- **000000004.12561@192.168.1.3 Mean:** 7.730769 **bids:** [51, 5, 5, 1, 32, 5, 5, 5, 5, 9, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 5, 5, 5, 5, 5, 5]
- **000000002.12561@192.168.1.3 Mean:** 7.9615383 **bids:** [40, 50, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 2, 5, 5, 5, 5, 5]
- **000000001.12561@192.168.1.3 Mean:** 9.0 **bids:** [18, 5, 5, 44, 5, 5, 5, 5, 5, 5, 5, 5, 5]
- **000000003.12561@192.168.1.3 Mean:** 7.888889 **bids:** [5, 5, 28, 5, 5, 5, 5, 5, 34, 5, 5, 5, 5, 5, 5, 5, 5, 5]

Master :: Total price: 667 **Average transaction price:** 13.34 **Simultaneous transaction count:** 24

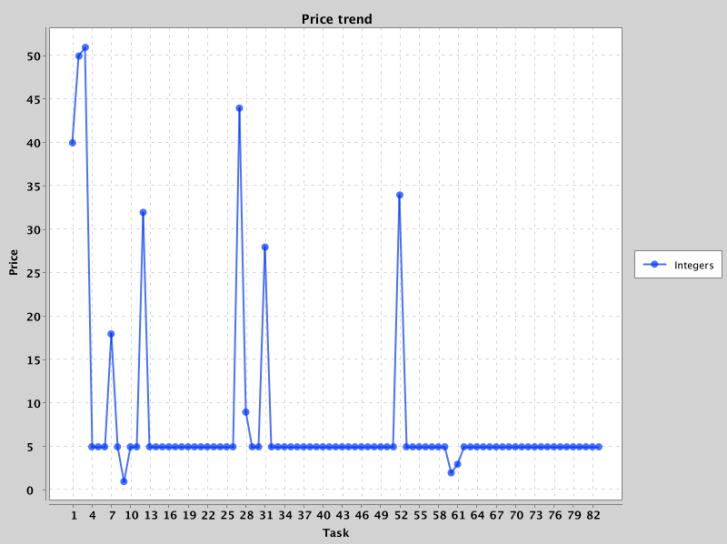

Figura 2.2: Andamento spese del Master con 4 worker con storage abilitato

2.1.3 Storage abilitato 3 Workers

Number of workers: 3 **Storage:** true **Execution Time:** 153.566 seconds

- **000000001.12676@192.168.1.3 Mean:** 7.6923075 **bids:** [69, 5, 7, 5, 5, 5, 5, 11, 5, 5, 5, 5, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
- **000000003.12676@192.168.1.3 Mean:** 8.347826 **bids:** [35, 5, 5, 5, 5, 5, 5, 5, 52, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
- **000000002.12676@192.168.1.3 Mean:** 8.466666 **bids:** [78, 5, 22, 19, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

Master :: Total price: 646 **Average transaction price:** 12.92 **Simultaneous transaction count:** 22

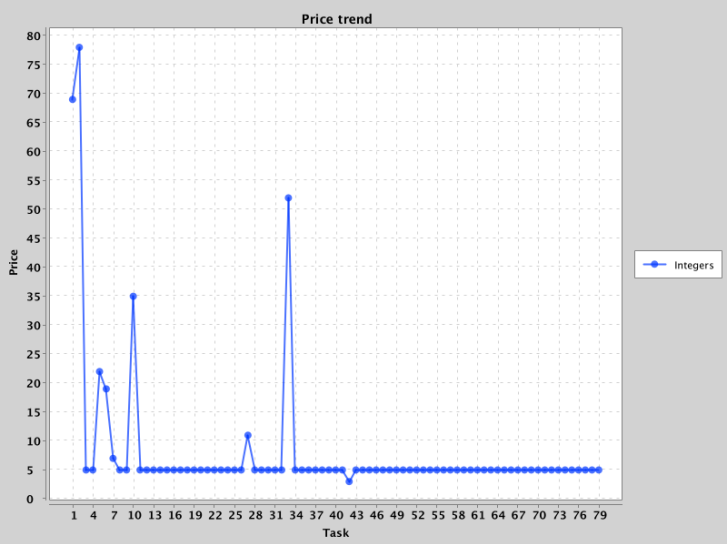

Figura 2.3: Andamento spese del Master con 3 worker con storage abilitato

2.1.4 Storage abilitato 2 Workers

Number of workers: 2 **Storage:** true **Execution Time:** 154.175 seconds

- **000000001.12790@192.168.1.3 Mean:** 7.6666665 **bids:** [90, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 21, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 0, 5, 5, 5]
- **000000002.12790@192.168.1.3 Mean:** 7.5666666 **bids:** [39, 12, 5, 5, 5, 5, 5, 35, 5, 11, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

Master :: Total price: 503 **Average transaction price:** 10.06 **Simultaneous transaction count:** 16

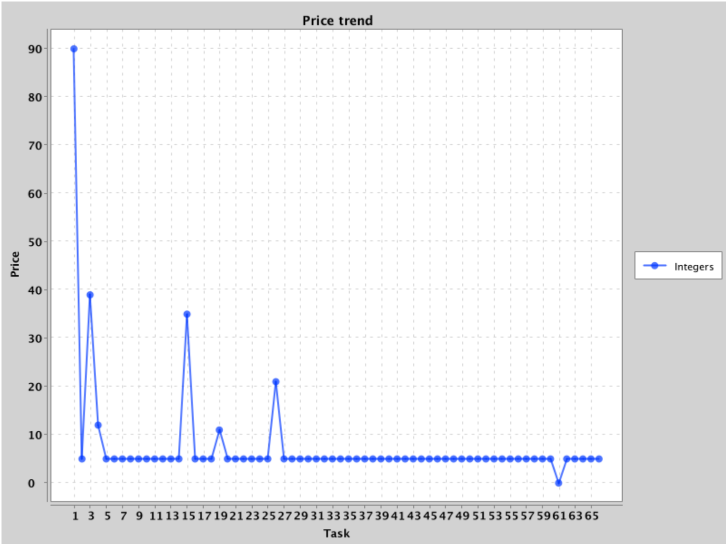

Figura 2.4: Andamento spese del Master con 2 worker con storage abilitato

2.1.5 Storage abilitato 1 Worker

Number of workers: 1 **Storage:** true **Execution Time:** 151.356 seconds

- **000000001.13747@192.168.1.3 Mean:** 7.18 **bids:** [116, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

Master :: Total price: 359 **Average transaction price:** 7.18 **Simultaneous transaction count:** 0

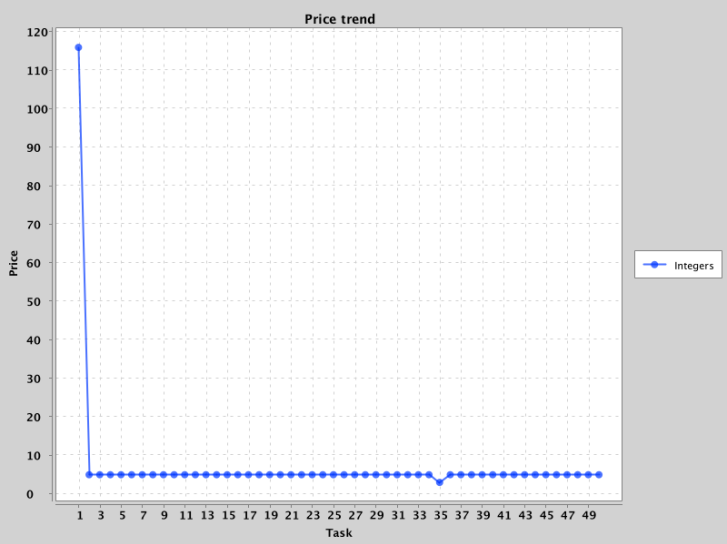

Figura 2.5: Andamento spese del Master con 1 worker con storage abilitato

2.1.6 Storage disabilitato 5 Workers

Number of workers: 5 **Storage:** false **Execution Time:** 154.206 seconds

- **000000001.13151@192.168.1.3 Mean:** 48.68 **bids:** [11, 65, 72, 90, 6, 21, 58, 30, 95, 2, 71, 66, 64, 42, 39, 81, 36, 30, 45, 92, 46, 43, 96, 7, 9]
- **000000005.13151@192.168.1.3 Mean:** 45.458332 **bids:** [31, 56, 6, 12, 30, 95, 13, 2, 51, 42, 77, 81, 36, 33, 52, 100, 46, 52, 67, 32, 96, 7, 65, 9]
- **000000002.13151@192.168.1.3 Mean:** 45.107143 **bids:** [18, 31, 49, 28, 11, 90, 6, 21, 58, 74, 95, 13, 2, 51, 64, 39, 81, 48, 36, 100, 45, 92, 46, 52, 32, 7, 65, 9]
- **000000004.13151@192.168.1.3 Mean:** 44.38095 **bids:** [47, 54, 49, 99, 11, 12, 29, 30, 95, 13, 51, 66, 36, 46, 52, 43, 96, 7, 65, 22, 9]

- **000000003.13151@192.168.1.3 Mean:** 48.86207 **bids:** [72, 90, 56, 6, 58, 12, 74, 30, 13, 2, 51, 66, 77, 81, 48, 36, 33, 47, 52, 30, 100, 45, 92, 32, 59, 43, 96, 7, 9]

Master :: Total price: 5920 **Average transaction price:** 118.4 **Simultaneous transaction count:** 38

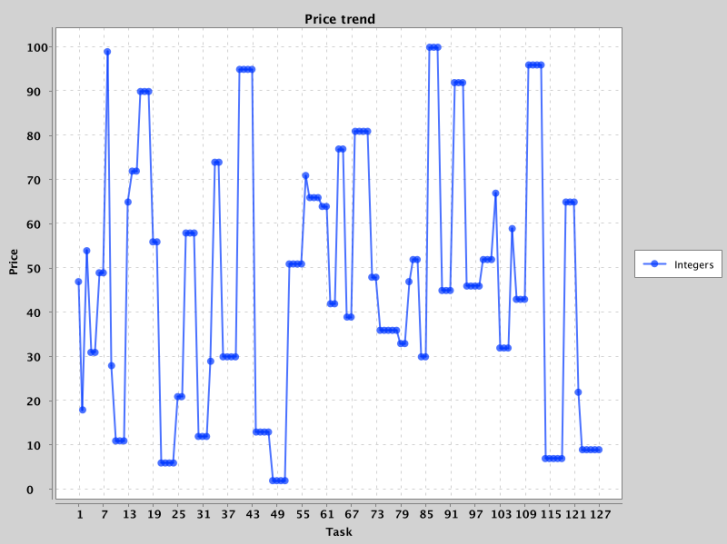

Figura 2.6: Andamento spese del Master con 5 worker con storage disabilitato

2.1.7 Storage disabilitato 4 Workers

Number of workers: 4 **Storage:** false **Execution Time:** 153.501 seconds

- **000000001.13285@192.168.1.3 Mean:** 40.411766 **bids:** [39, 32, 38, 14, 6, 47, 97, 2, 13, 6, 9, 28, 78, 35, 52, 97, 94]
- **000000003.13285@192.168.1.3 Mean:** 42.88 **bids:** [20, 23, 38, 94, 92, 38, 70, 14, 6, 28, 61, 21, 44, 4, 70, 53, 6, 66, 45, 28, 4, 23, 78, 52, 94]
- **000000002.13285@192.168.1.3 Mean:** 47.61905 **bids:** [26, 48, 38, 92, 38, 70, 14, 6, 43, 21, 44, 12, 81, 71, 13, 66, 4, 89, 52, 98, 74]

- **000000004.13285@192.168.1.3 Mean:** 50.8 **bids:** [10, 48, 94, 53, 43, 32, 62, 55, 44, 97, 12, 35, 4, 2, 71, 88, 23, 52, 97, 94]

Master :: Total price: 3775 **Average transaction price:** 75.5 **Simultaneous transaction count:** 26

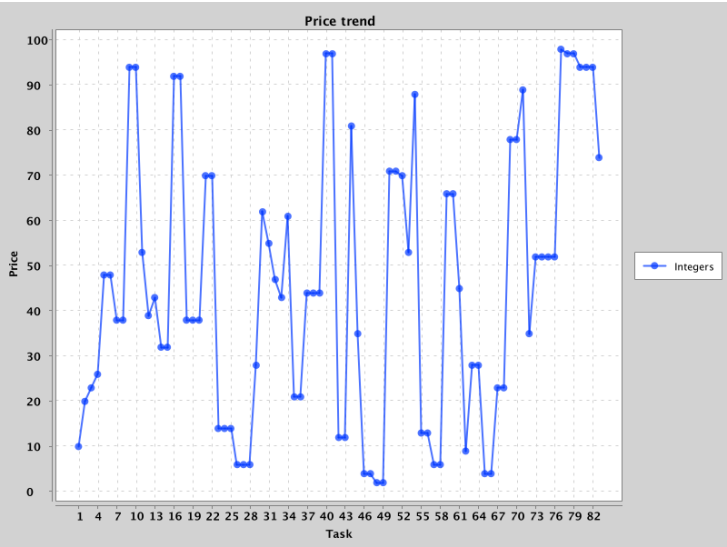

Figura 2.7: Andamento spese del Master con 4 worker con storage disabilitato

2.1.8 Storage disabilitato 3 Workers

Number of workers: 3 **Storage:** false **Execution Time:** 151.83 seconds

- **000000003.13402@192.168.1.3 Mean:** 49.0 **bids:** [49, 5, 25, 87, 59, 23, 11, 87, 56, 72, 75, 24, 19, 92, 80, 60, 5, 71, 76, 80, 78, 45, 16, 11, 47, 21]

- **000000001.13402@192.168.1.3 Mean:** 47.885715 **bids:** [93, 11, 79, 25, 87, 42, 59, 26, 17, 23, 11, 53, 71, 63, 56, 72, 89, 47, 24, 19, 87, 30, 92, 15, 80, 5, 71, 80, 78, 11, 44, 9, 37, 59, 11]

- **000000002.13402@192.168.1.3 Mean:** 47.05 **bids:** [93, 11, 25, 98, 26, 30, 23, 61, 72, 75, 24, 34, 71, 71, 80, 11, 47, 9, 21, 59]

Master :: Total price: 3891 **Average transaction price:** 77.82 **Simultaneous transaction count:** 25

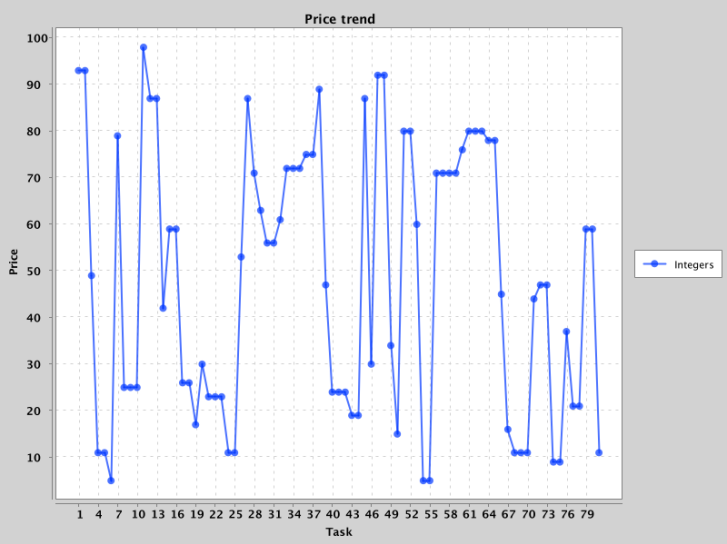

Figura 2.8: Andamento spese del Master con 3 worker con storage disabilitato

2.1.9 Storage disabilitato 2 Workers

Number of workers: 2 **Storage:** false **Execution Time:** 151.066 seconds

- **000000002.13516@192.168.1.3 Mean:** 46.322582 **bids:** [84, 78, 37, 59, 10, 42, 80, 72, 16, 82, 100, 14, 53, 2, 85, 43, 53, 3, 33, 40, 35, 5, 17, 40, 31, 54, 21, 92, 10, 84, 61]
- **000000001.13516@192.168.1.3 Mean:** 48.303032 **bids:** [55, 49, 72, 64, 36, 19, 30, 84, 59, 29, 35, 76, 80, 72, 68, 82, 50, 53, 2, 59, 85, 3, 14, 40, 49, 17, 59, 54, 21, 11, 7, 76, 84]

Master :: Total price: 3030 **Average transaction price:** 60.6 **Simultaneous transaction count:** 14

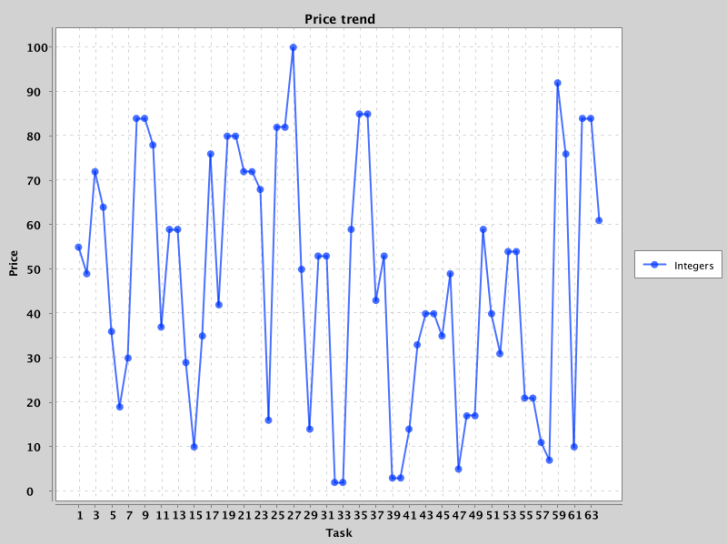

Figura 2.9: Andamento spese del Master con 2 worker con storage disabilitato

2.1.10 Storage disabilitato 1 Worker

Number of workers: 1 **Storage:** false **Execution Time:** 152.336 seconds

- **000000001.13639@192.168.1.3 Mean:** 45.82 **bids:** [40, 87, 12, 16, 19, 19, 14, 26, 92, 41, 51, 17, 32, 7, 18, 68, 70, 75, 25, 44, 71, 61, 36, 26, 38, 72, 49, 90, 27, 83, 29, 70, 36, 28, 54, 30, 13, 68, 36, 52, 34, 2, 78, 28, 59, 49, 80, 85, 59, 75]

Master :: Total price: 2291 **Average transaction price:** 45.82 **Simultaneous transaction count:** 0

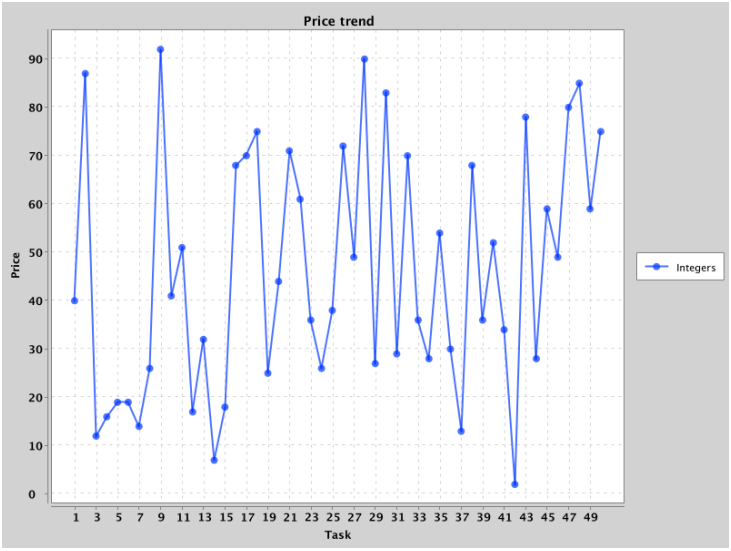

Figura 2.10: Andamento spese del Master con 1 worker con storage disabilitato

Capitolo 3

Guida all'installazione

Il progetto può essere clonato dalla repository GitHub:
`https://github.com/alextivoli/ContractNet/tree/main` oppure è possibile scaricare lo zip da estrarre.
Come si vede dalla figura 3.1 nella repository sono presenti due package:

- actodes, che contiene il software actodes non modificato
- contractnet, che contiene il codice del progetto

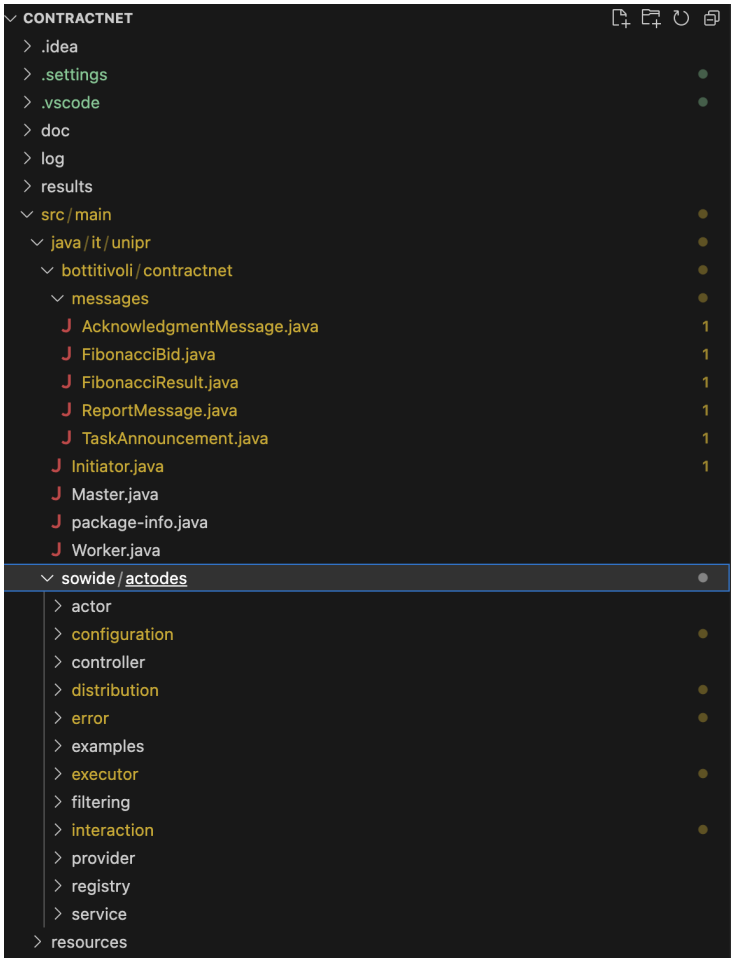

Figura 3.1: Struttura della repository

3.1 Dipendenze esterne

Il progetto si basa su ActoDes, nella repository Github ActoDes è **incluso** e **non è stato modificato**.
È necessaria una dipendenza esterna per il progetto proposto: **XChart**.
XChart è una libreria che permette di creare dei grafici, nel progetto è stata inserita al fine di creare i grafici dell'andamento delle spese del Master.

```
<dependency>
    <groupId>org.knowm.xchart</groupId>
    <artifactId>xchart</artifactId>
    <version>3.8.0</version>
</dependency>
```

Nel file pom.xml presente nella repository è già inclusa e non occorre aggiungerla manualmente. Nel caso non si stesse lavorando sulla repository fornita e si volesse aggiungere manualmente occorre aggiungere la dipendenza tramite Maven andando a modificare il pom.xml.

3.1.1 Esecuzione del codice

Per eseguire il codice occorre eseguire la classe Initiator senza nessun argomento. All'avvio verrà chiesto tramite input da tastiera il numero dei workers (inserire un numero tra 1 e 5) e se lo storage è abilitato o meno (digitare y per abilitare, n per non abilitare). Dopodichè il sistema comincerà il funzionamento e sarà possibile osservare a console le varie stampe presenti.
Finita l'esecuzione sarà possibile visualizzare il report dei risultati nel file output.txt presente nella cartella *results* della repository. Oltre al report sarà generata un'immagine .png contenente il trend delle spese del Master, l'immagine assumerà il seguente nome: *NBool.png* dove *N* indica il numero dei workers presenti nell'esecuzione, mentre *Bool* indica se lo storage è abilitato oppure no.