Asta distribuita

Vincenzo Luigi Bruno & Salvatore Cangiano

2024

1 Abstract

Questo progetto presenta un sistema di asta distribuita basato su architettura IoT, in cui un algoritmo distribuito garantisce l'ordinamento totale e causale delle offerte dei partecipanti. Il sistema utilizza un Sequenziatore per l'ordinamento totale, che è responsabile di assegnare un ordine globale agli eventi dell'asta, e si affida ai Vector Clocks per mantenere la causalità tra gli eventi distribuiti. Il Sequenziatore riveste un ruolo duplice di direttore e partecipante dell'asta, mentre gli altri nodi agiscono come semplici partecipanti

2 Componenti Hardware

Nel progetto sono stati utilizzati i seguenti componenti hardware:

- 5 microcontrollori ESP-32
- 5 LCD
- 6 push-button
- Breadboard
- Connettori
- 6 resistenze da 5KΩ

3 Architettura

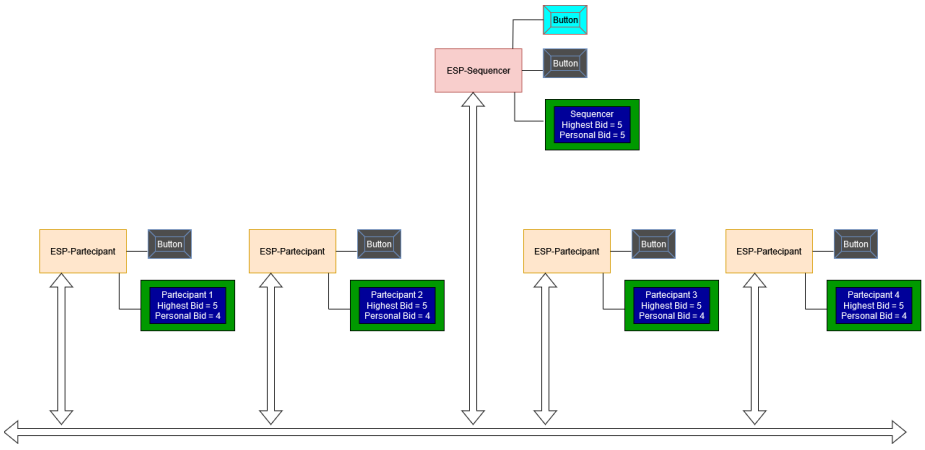
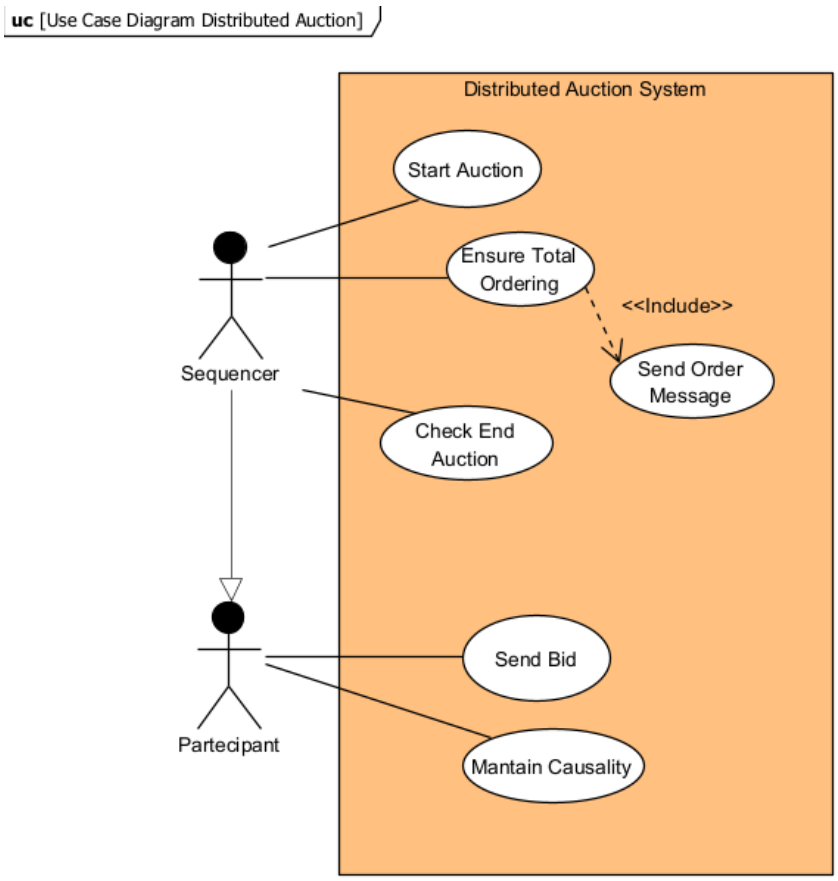


Figure 1: Schema architetturale con componenti hardware

4 Diagramma dei Casi d'Uso



Il diagramma rappresenta il sistema di asta distribuita, dove i ruoli principali sono svolti da due attori: **Sequenziatore** e **Partecipante**. Il Sequenziatore è una *specializzazione* del Partecipante, poiché eredita le sue funzionalità di base e ne aggiunge altre specifiche per garantire il corretto funzionamento del sistema.

4.1 Attore: Partecipante

I suoi casi d'uso principali sono:

- **Inviare Offerta (*Send Bid*)**: Un partecipante può inviare un'offerta al sistema di asta.
- **Mantenere la Causalità (*Maintain Causality*)**: I partecipanti collaborano per garantire la coerenza causale tra gli eventi distribuiti tramite

i Vector Clocks.

4.2 Attore: Sequenziatore

È un partecipante speciale che, oltre a eseguire le attività di base di un partecipante, ha responsabilità aggiuntive per il corretto funzionamento dell'asta.

I suoi casi d'uso principali sono:

- **Avviare l'Asta (*Start Auction*)**: Il Sequenziatore può inizializzare un'asta, configurandone i parametri.
- **Garantire l'Ordinamento Totale (*Ensure Total Ordering*)**: Assegna un timestamp globale agli eventi per garantire l'ordinamento totale. **Include** l'azione di *Send Order Message*, dove il Sequenziatore invia ai partecipanti i messaggi d'ordine per notificare l'ordine stabilito.
- ***Send Order Message***: Caso d'uso incluso in Ensure Total Ordering. Il Sequenziatore notifica ai partecipanti l'ordine globale delle offerte.
- **Verificare la Fine dell'Asta (*Check End Auction*)**: Il Sequenziatore verifica la fine dell'asta e determina il vincitore in base alle offerte.

5 Assunzioni

Il progetto si basa su alcune assunzioni iniziali per la realizzazione del sistema di asta distribuita, che semplificano l'implementazione dell'algoritmo distribuito e garantiscono il corretto funzionamento del Sequenziatore e dei partecipanti. Le principali assunzioni sono:

1. **Gruppi chiusi non sovrapposti**: I partecipanti al sistema sono organizzati in gruppi definiti, e non vi è sovrapposizione tra i membri dei gruppi. Questa struttura è fondamentale per ridurre la complessità della gestione dei *Vector Clocks*, che dipendono direttamente dalla dimensione del gruppo n.
2. **Numero di processi N (con $N \leq 5$)**:

   Sono presenti N processi distribuiti nel sistema. Questa assunzione implica che tutti i nodi siano noti a priori e possano collaborare per mantenere l'integrità del sistema.

   Se il numero di partecipanti per gruppo è relativamente piccolo, il vettore dei timestamp (che deve contenere un valore per ogni membro del gruppo) sarà più leggero e l'aggiornamento dei Vector Clocks sarà più efficiente.
3. **Sequenziatore interno**: Il Sequenziatore non è un'entità separata dal gruppo, ma parte integrante dello stesso. Agisce sia come un normale partecipante, che invia e riceve messaggi, sia come un direttore che assegna i numeri di sequenza per garantire l'ordinamento totale.

4. **Canali di comunicazione affidabili**: Esistenza di canali di comunicazione affidabili, dove i messaggi inviati arrivano sempre correttamente ai destinatari.

5. **Comunicazione broadcast in reti ad-hoc**: I nodi devono collaborare per garantire la propagazione dei messaggi in assenza di una rete centralizzata.

6. **Processi non falliscano**: Si assume che i processi coinvolti nel sistema, inclusi sia i nodi partecipanti che il sequenziatore, siano sempre attivi e operativi

6 Modello del Sistema

Il sistema si compone di 5 processi, nei quali ci sarannno 4 partecipanti semplici ed un sequenziatore. Il sequenziatore fa parte del gruppo e partecipa attivamente all'asta, inviando anche lui delle offerte. Ogni membro del gruppo ha a disposizione queste variabili globali:

- sequenceNumber
- vectorClock
- highestBid
- nodeId
- auctionIsStarted

1. Il vectorClock serve ai membri del gruppo a far mantenere un ordine causale dei messaggi, verrà aggiornato correttamente all'invio di un messaggio di offerta e alla ricezione causale di altri messaggi.

2. Il sequenceNumber serve ai membri del gruppo per tenere un ordinamento totale dei messaggi ricevuti, verrà aggiornato ogni qual volta si riceve un messaggio di ordinamento con il sequenceNumber attuale.

3. HighestBid serve per tenere traccia dell'offerta proposta più alta, se un membro vuole fare un'offerta è costretto ad aumentare l'HighestBid corrente.

4. NodeId tiene traccia dell'id del nodo, utile per aggiungere questa informazione ai messaggi che il membro invierà al gruppo.

5. AuctionIsStarted è una variabile booleana, serve a far capire ai membri quando è in corso un'asta o meno.

7 Algoritmo

7.1 Struttura del messaggio

Il messaggio è fornito di un campo **senderId** che determina il mittente del messaggio. Inoltre ha anche un **messageId** che è un identificativo univoco nel contesto del mittente, la coppia **senderId** e **messageId** creano un identificativo univoco per ogni messaggio nel sistema.

Il messaggio ha un campo che ne indica il tipo: in base al tipo di messaggio i partecipanti si comporteranno in maniera differente. I tipi di messaggio sono quattro, ossia **bid**, **order**, **start**, **end**.

Altri campi utili sono **bidValue**, **sequenceNumber** e **vectorClock**. Il campo **bidValue** è il valore dell'offerta che è trasportata dal messaggio, chiunque riceva un messaggio di tipo **bid** utilizzerà questo campo per confrontare la propria variabile interna **highestBid** ed aggiornarla di conseguenza. **VectorClock** è il vettore logico del mittente al momento dell'invio, **sequenceNumber** è un campo valido solo per i messaggi di ordinamento e corrisponde al numero di sequenza utilizzato dai partecipante per ordinare i messaggi.

In base al tipo altri campi del messaggio possono essere invalidati o meno, ad esempio un messaggio di tipo **start** avrà invalidato il campo **bid**.

Ecco riportata la struttura del messaggio nel dettaglio:

Struct Message
- **Campo** type : Stringa
- **Campo** Bid : Intero
- **Campo** SequenceNumber : Intero
- **Campo** VectorClock : Array[Intero]
- **Campo** senderId : Intero
- **Campo** MessageId : Intero

7.2 Ciclo indefinito

7.2.1 Ciclo Sequenziatore

```
while true do
    if checkButtonAuctionStarted() AND !auctionIsStarted then
        startAuction()
        sendStartAuction()
    end if
    if auctionIsStarted then
        checkEndAuction()
        if checkButtonBid() then
            sendBid()
        end if
```

```
    end if
end while
```

7.3 Ciclo Partecipante

```
while true do
    if auctionIsStarted then
        checkEndAuction()
        if checkButtonBid() then
            sendBid()
        end if
    end if
end while
```

7.4 Ricezione di un messaggio

I messaggi che i partecipanti possono ricevere sono di vario tipo. Il partecipante può ricevere un messaggio di inizio asta e di fine asta, servono rispettivamente ad inizializzare le variabili e le code utilizzate per gestire l'ordinamento causale e totale, e a concludere la ricezione dei messaggi.

Può ricevere dei messaggi di offerta, alla loro ricezione il partecipante controllerà se il messaggio è "causale". Se non è "causale" verrà messo in una coda di attesa **HoldBackQueue**, altrimenti viene spostato in un altra coda **CausalQueue** in attesa del corrispettivo messaggio di ordinamento.

Infine il partecipante può ricevere un messaggio di ordinamento, quest'ultimo serve a stabilire un ordine unico per i messaggi nella coda **CausalQueue**. Se all'arrivo di un messaggio di ordinamento, nella coda **CausalQueue** non è presente il suo corrispettivo, il messaggio di ordinamento verrà messo nella coda **OrderQueue**.

7.4.1 Ricezione Partecipante

```
When a message arrives
type ← message.type
if type = start then
    startAuction()
else if type = end then
    endAuction()
else if type = bid then
    push message in HoldBackQueue
    repeat
        thereIsCausalMessage ← false
        for message in HoldBackQueue do
            if causalControl(message) then
```

```
                thereIsCausalMessage ← true
                pop message from HoldBackQueue
                push message in CausalQueue
                if orderControl(message) then
                    TO-Deliver()
                    pop messages from CausalQueue and OrderQueues
                end if
                break
            end if
        end for
    until thereIsCausalMessage = true
else if type = order then
    push message in OrderQueue
    repeat
        thereIsOrderMessage ← false
        for message in OrderQueue do
            if orderControl(message) then
                thereIsOrderMessage ← true
                TO-Deliver()
                pop message from OrderQueue and CausalQueue
                break
            end if
        end for
    until thereIsOrderMessage = true
end if
```

Il sequenziatore quando riceve un messaggio si comporta diversamente. Difatti il sequenziatore scarterà i messaggi di ordinamento inviati da se stesso in quanto ha già fatto la TO-Deliver. Il sequenziatore reagirà solamente ai messaggi di offerta, e si comporterà in modo simile ad un partecipante. Quando però farà il controllo causale del messaggio, invierà direttamente il messaggio di tipo order a tutti i partecipanti.

7.4.2 Ricezione Sequenziatore

```
When a message arrives
type ← message.type
if type = bid then
    push message in HoldBackQueue
    repeat
        thereIsCausalMessage ← false
        for message in HoldBackQueue do
            if causalControl(message) then
                thereIsCausalMessage ← true
                pop message from HoldBackQueue
```

```
            TO-Send(message)
            break
        end if
    end for
  until thereIsCausalMessage = true
else if type = order OR type = start OR type = end then
  Do Nothing
end if
```

8 Problemi

Di seguito vengono descritti i principali problemi che potrebbero verificarsi durante l'operazione del sistema, con un focus particolare su fallimenti, colli di bottiglia, e problemi di comunicazione, come la perdita o duplicazione dei messaggi.

1. **Il Sequenziatore è un Single Point of Failure:**
   - **Scenario**: il Sequenziatore fallisce.
   - **Effetti**: In assenza del Sequenziatore, i partecipanti non possono più ricevere numeri di sequenza per i loro messaggi e quindi non possono proseguire con l'invio delle offerte o altre azioni previste dal sistema.
   - **Impatto**: L'asta si interrompe, non potendo più proseguire con la gestione delle offerte.
2. **Il Sequenziatore è un Bottleneck**:
   - **Scenario**: il numero di partecipanti aumenta o la frequenza degli eventi cresce.
   - **Effetti**: La latenza nell'assegnazione dei numeri di sequenza aumenta, rallentando l'intero processo di gestione delle offerte e degli eventi. I partecipanti potrebbero dover attendere più tempo per ricevere la conferma del numero di sequenza, rallentando l'interazione con il sistema.
   - **Impatto**: In un contesto con un numero crescente di partecipanti o una maggiore frequenza di offerte, il rallentamento del Sequenziatore potrebbe compromettere l'efficienza dell'asta, creando ritardi significativi e impedendo una gestione tempestiva delle offerte.
3. **Messaggi persi:**
   - **Scenario**: Un messaggio da parte del Sequenziatore non viene ricevuto da uno o più partecipanti a causa della perdita di pacchetti nella rete.

- **Effetti**: Alcuni partecipanti ricevono i numeri di sequenza, mentre altri no, causando inconsistenze nell'ordinamento globale.
- **Impatto**: I nodi con messaggi mancanti potrebbero generare eventi fuori ordine o bloccare l'elaborazione.

4. **Messaggi duplicati:**

   - **Scenario**: Un messaggio viene duplicato nella rete.
   - **Effetti**: I partecipanti ed il Sequenziatore potrebbero processare più volte lo stesso messaggio.
   - **Impatto**: Potrebbero verificarsi offerte duplicate e id dei messaggi incoerenti.