Prima prova pratica in itinere: controller ABS con diagnostica gestita da un PS

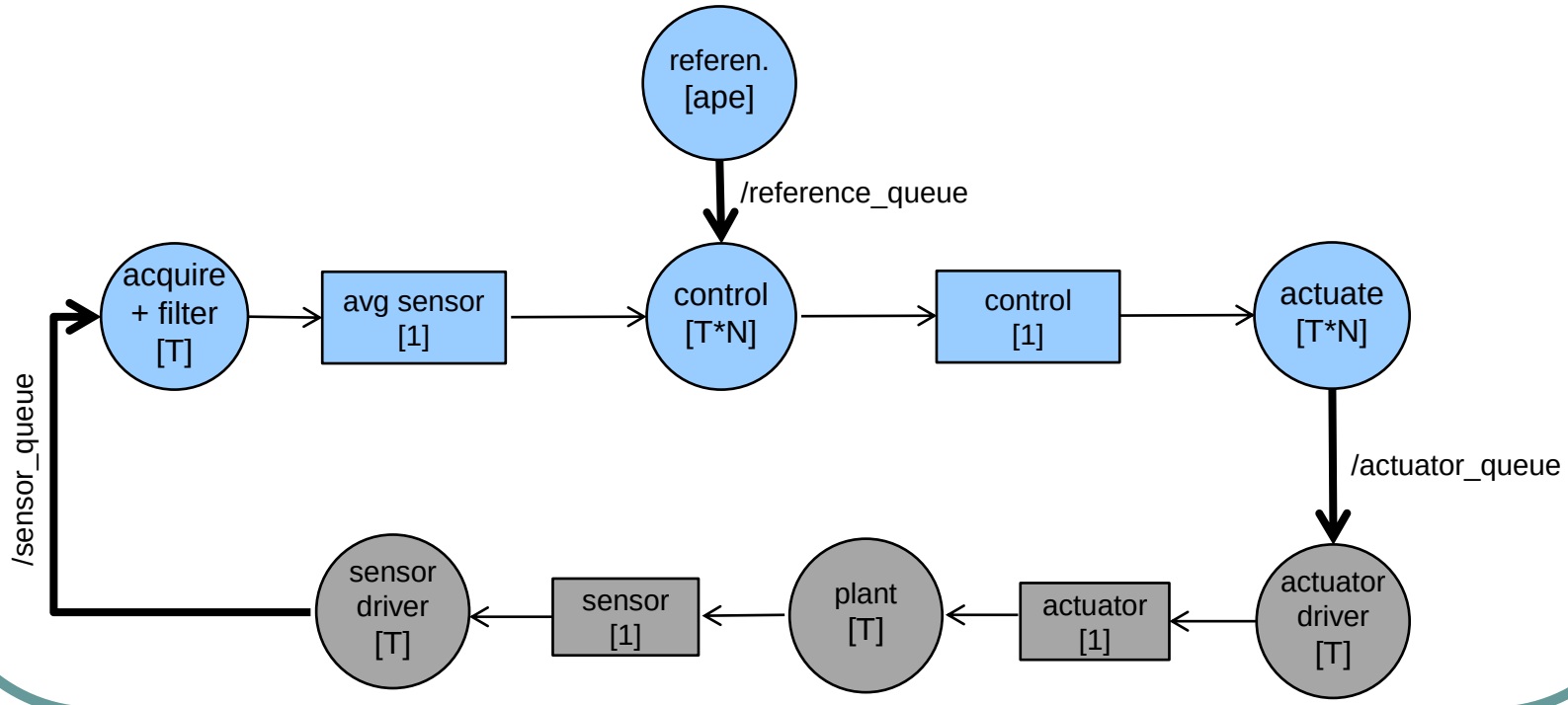
Corso di
Progetto e Sviluppo di
Sistemi in Tempo Reale
a.a. 2022/23

Marcello Cinque

Schema di partenza: loop di controllo

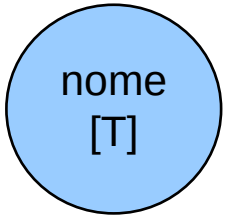
- Una tipica applicazione di controllo prevede più stadi di processing, realizzati da altrettanti task periodici:
  - Acquisizione dati dai sensori + filtraggio (ad es., rimozione rumore)
  - Elaborazione legge di controllo
  - Invio dati agli attuatori
- Ognuno di tali task può essere replicato per quante sono le grandezze da controllare nell'impianto
- Inoltre, task accessori (anche aperiodici) possono essere aggiunti per monitoraggio, diagnostica, …
- La sincronizzazione e comunicazione tra i task può essere realizzata con le primitive POSIX illustrate

Schema di partenza: loop di controllo

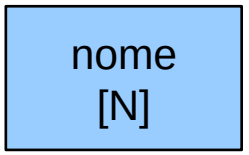


Simbologia dello schema

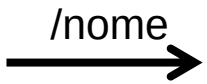
- I cerchi rappresentano i task
  - Il valore tra parentesi quadre rappresenta il periodo del task, o “ape” per i task aperiodici



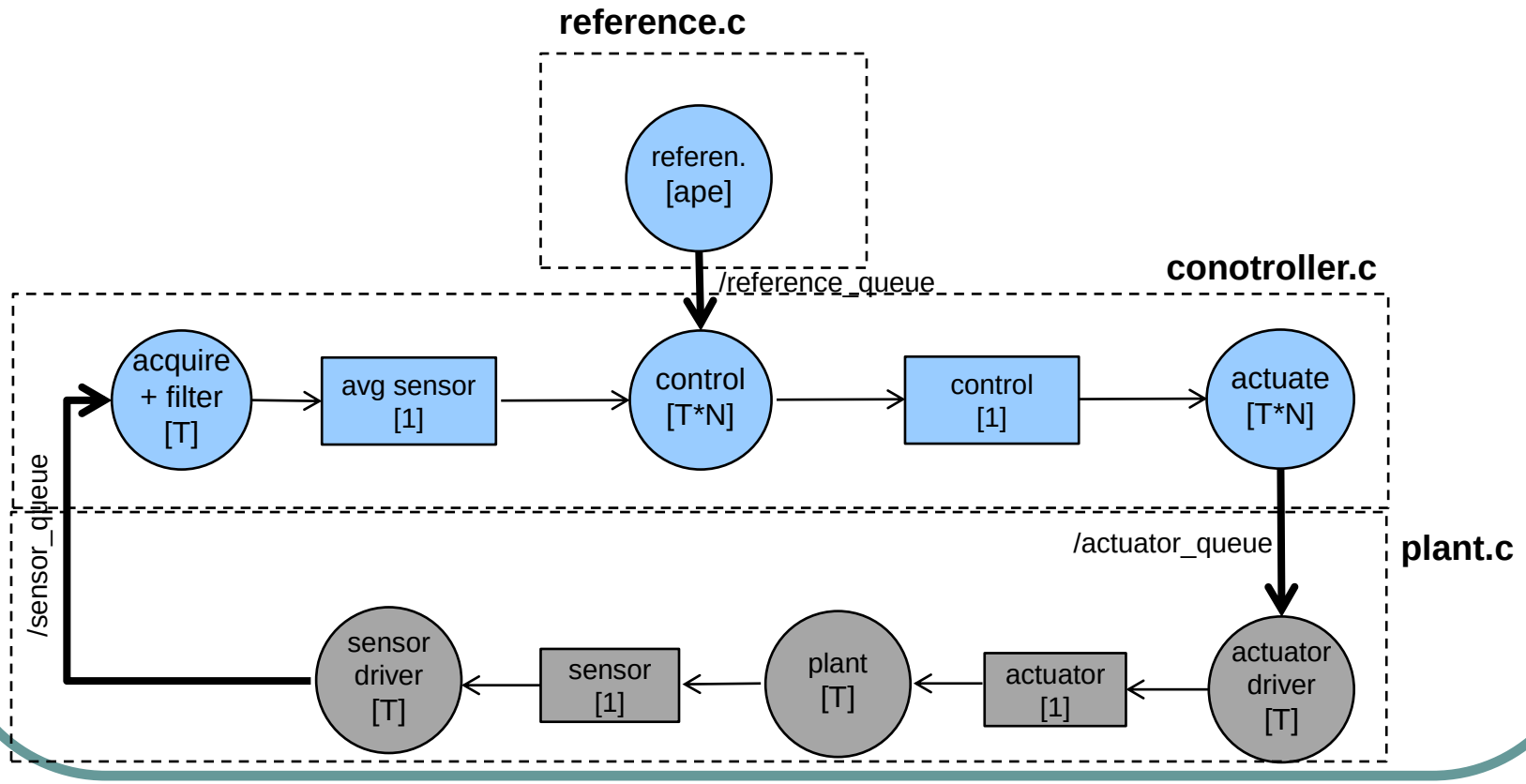
- I rettangoli rappresentano le risorse condivise
  - Il valore tra parentesi quadre rappresenta la dimensione della risorsa
  - Le risorse si intendono protette da mutex



- Le frecce dirette tra task rappresentano messaggi con comunicazione tramite code



Schema di partenza: mapping sui file sorgente



Il plant

- Il plant simula la velocità di rotazione di una ruota normalmente in decelerazione in piano
- L'impianto accetta in ingresso il segnale di attuazione di accelerazione, decelerazione, frenata o nessuna attraverso la /actuator_queue
- L'impianto scrive il valore corrente della velocità di rotazione corrente sulla /sensor_queue

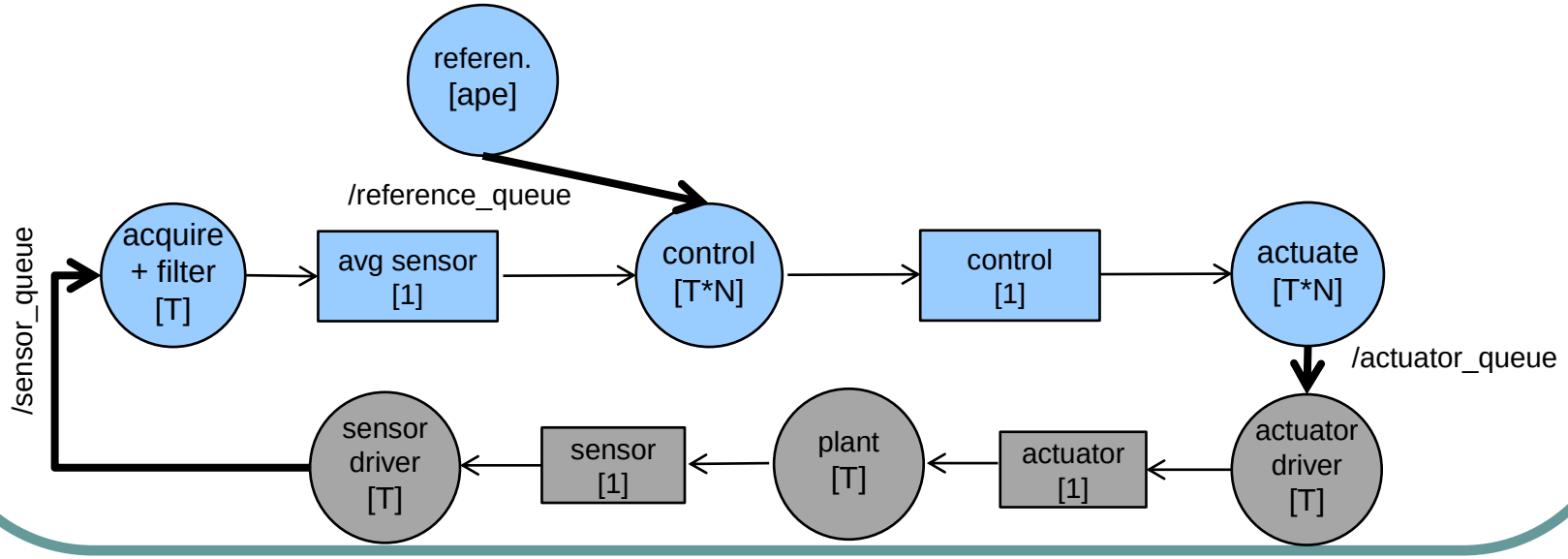
Il controllore

- Il controller acquisisce la velocità corrente, la confronta con il valore desiderato (impostato dal reference) e determina l'azione da effettuare
- I comandi che il controller invia all'attuatore sono codificati come segue
  - 0: nessuna azione
  - +1: accelerazione
  - -1: decelerazione
  - -2: frenata
- assumendo che un valore di reference pari a 0 equivalga al comando di frenata
  - ./reference 0 → FRENATA! (actuator=-2 sulle ruote)

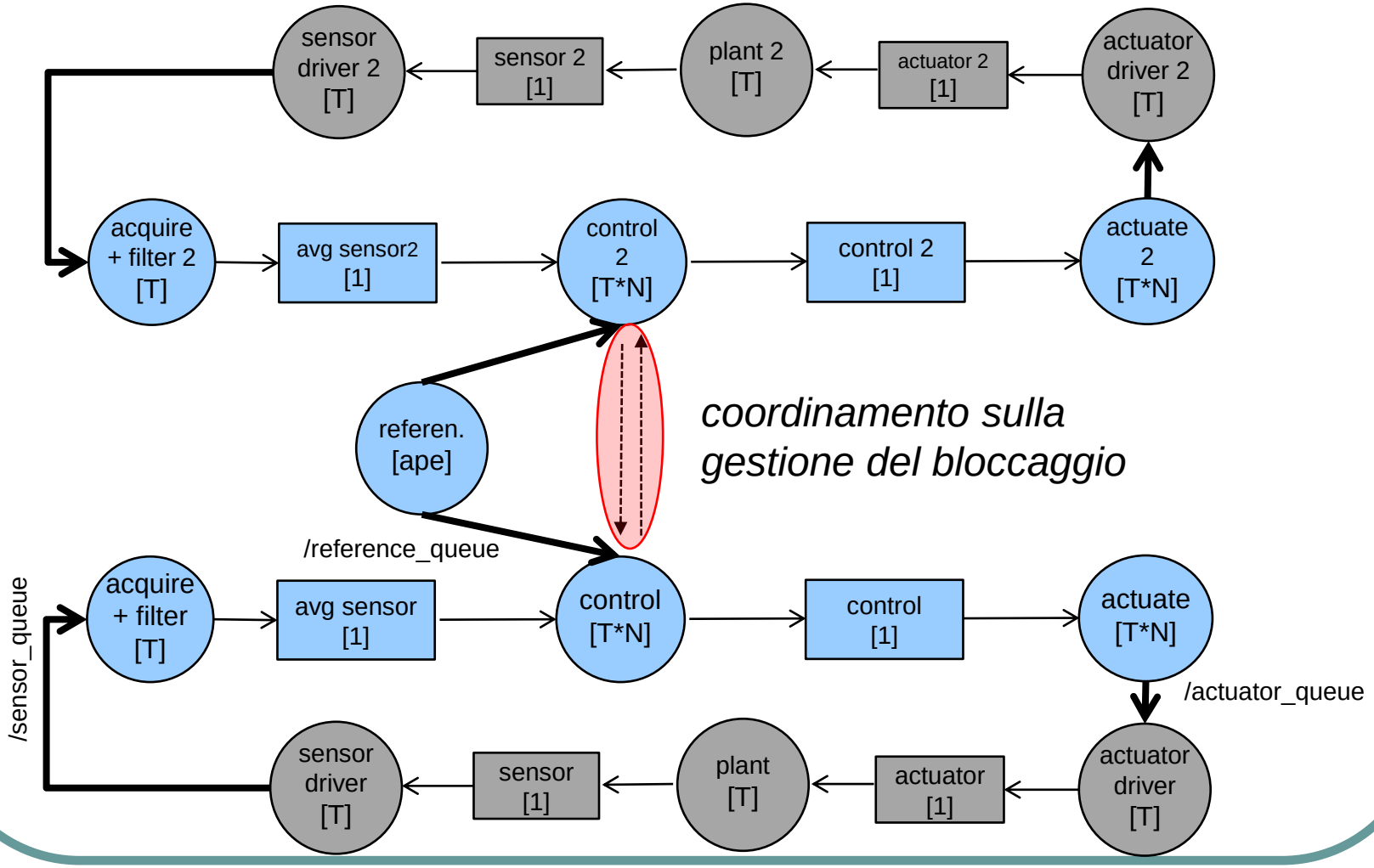
Traccia

- Si vogliono controllare due ruote simultaneamente (duplicando l'impianto)
- La frenata può generare lo slittamento (skating) delle ruote, indipendentemente l'una dall'altra
- I due controllori devono evitare lo skating (abs) e devono coordinarsi tra loro nel controllo simultaneo delle due ruote
- Si vuole inoltre dotare il controllore di un sistema di diagnostica asincrona gestito da un *Polling Server (PS)*

Schema AS IS


referen.
[ape]
/reference_queue
/sensor_queue
acquire
+ filter
[T]
avg sensor
[1]
control
[T*N]
control
[1]
actuate
[T*N]
/actuator_queue
sensor
driver
[T]
sensor
[1]
plant
[T]
actuator
[1]
actuator
driver
[T]

Schema TO BE: controllo coordinato



Note

- L'implementazione dei blocchi in grigio (plant) e del controller di partenza è fornita in HW1-ABS.zip
- Rispetto all'implementazione di partenza:
  - Bisogna duplicare il plant per simulare la seconda ruota
  - Bisogna duplicare il controllore e i due controller devono coordinarsi tra loro nel caso di bloccaggio di una delle ruote:
    - Se una ruota si blocca, il freno va rilasciato per qualche ciclo
    - Anche il controllore dell'altra ruota deve fare lo stesso
- Il coordinamento tra i due controller può essere fatto con scambio messaggi o con una memoria comune, a seconda se i due cicli di controllo sono implementati nello stesso sorgente o in due sorgenti separati

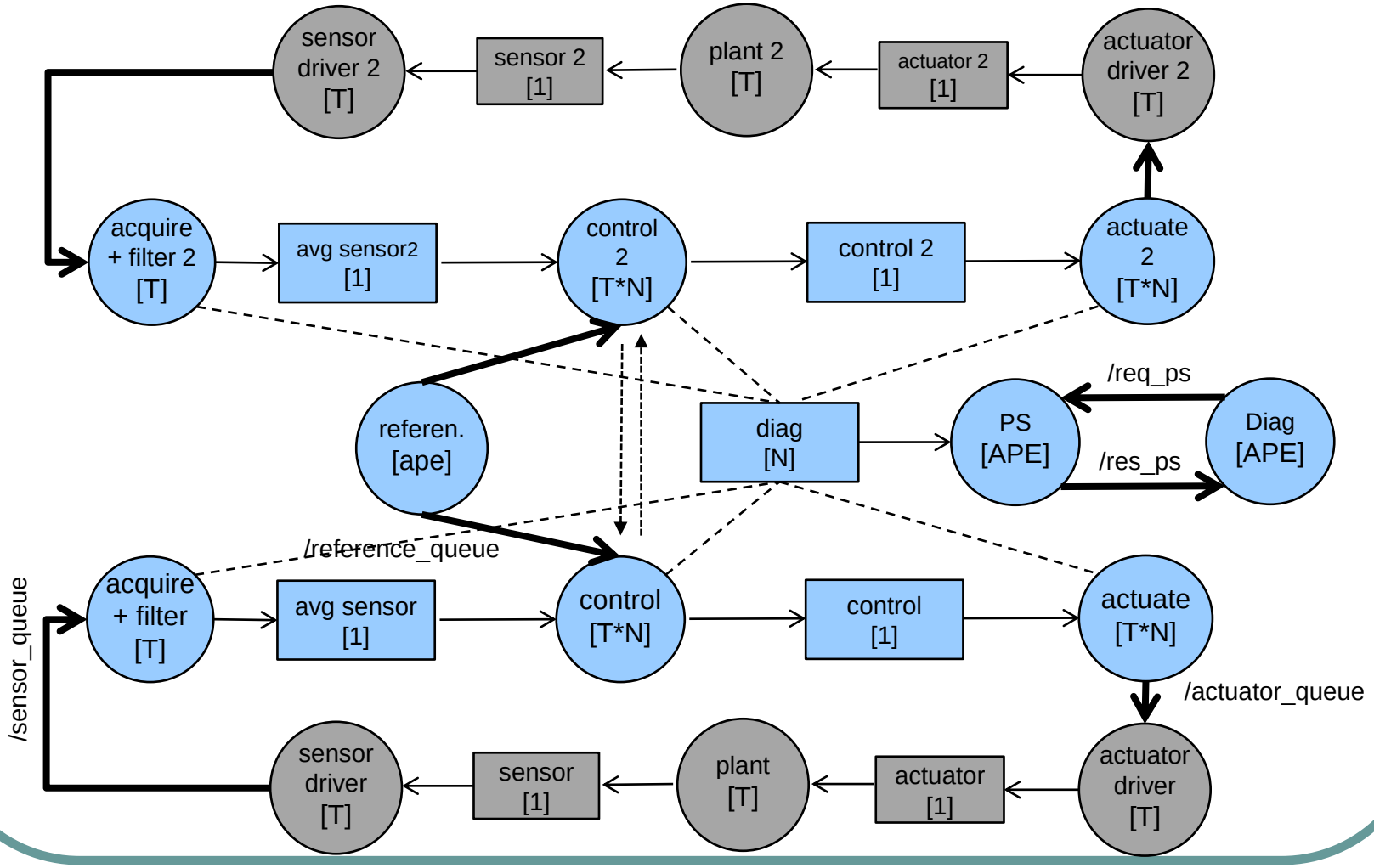
Diagnostica con Polling Server

- Ogni task salva in un area condivisa, acceduta **in mutua esclusione** evitando fenomeni di inversione di priorità
  - Il proprio WCET, misurato a runtime con la primitiva **clock_gettime**
  - Eventuali dati utili alla diagnostica (ad es, il segnale di controllo corrente, o il contenuto corrente del buffer)
- Un task esterno aperiodico «Diag» può chiedere di accedere e leggere le informazioni di diagnostica
- La richiesta viene raccolta ed eseguita da uno Polling Server (PS) che accede all'area condivisa in mutua esclusione e risponde a Diag

Diagnostica con Polling Server

- Il task diag, quando avviato, invia una richiesta di diagnostica al controller tramite la coda /req_ps
- Il PS, quando riceve una richiesta sulla coda /req_ps
  - Preleva i valori delle variabili presenti nella struttura condivisa, ad es
    - WCET dei task
    - Avg_sensor
    - control
    - buffer (gli elementi elementi contenuti nel buffer del task acquire_filter)
    - la reference
  - Li invia al task diag sulla coda /res_ps
- Quando il task diag riceve la risposta, stampa lo stato del controllore a video e termina
- Per semplicità si suppone che la richiesta venga evasa entro la capacità del PS e che non possano esserci più richieste simultanee

Schema TO BE completo



Note

- Se i controller sono implementati in file diversi, ci dovranno essere due task PS e il diag dovrà inviare la richiesta ad entrambi
- Tutti i task periodici devono essere schedulati con Rate Monotonic
- E’ opportuno dunque che eseguano tutti sulla stessa CPU (impostando l’affinity o utilizzando il comando taskset)

Consegna

- Entro le 20:00 del 12 maggio 2022
- E’ possibile lavorare in gruppi di massimo 4 persone
- Ogni studente deve consegnare individualmente l’elaborato su Teams, aggiungendo, se ha lavorato in gruppo, un file di testo con i nominativi e matricole dei componenti del gruppo