Esercizio 2: TMR e diagnosi con Deferrable Server

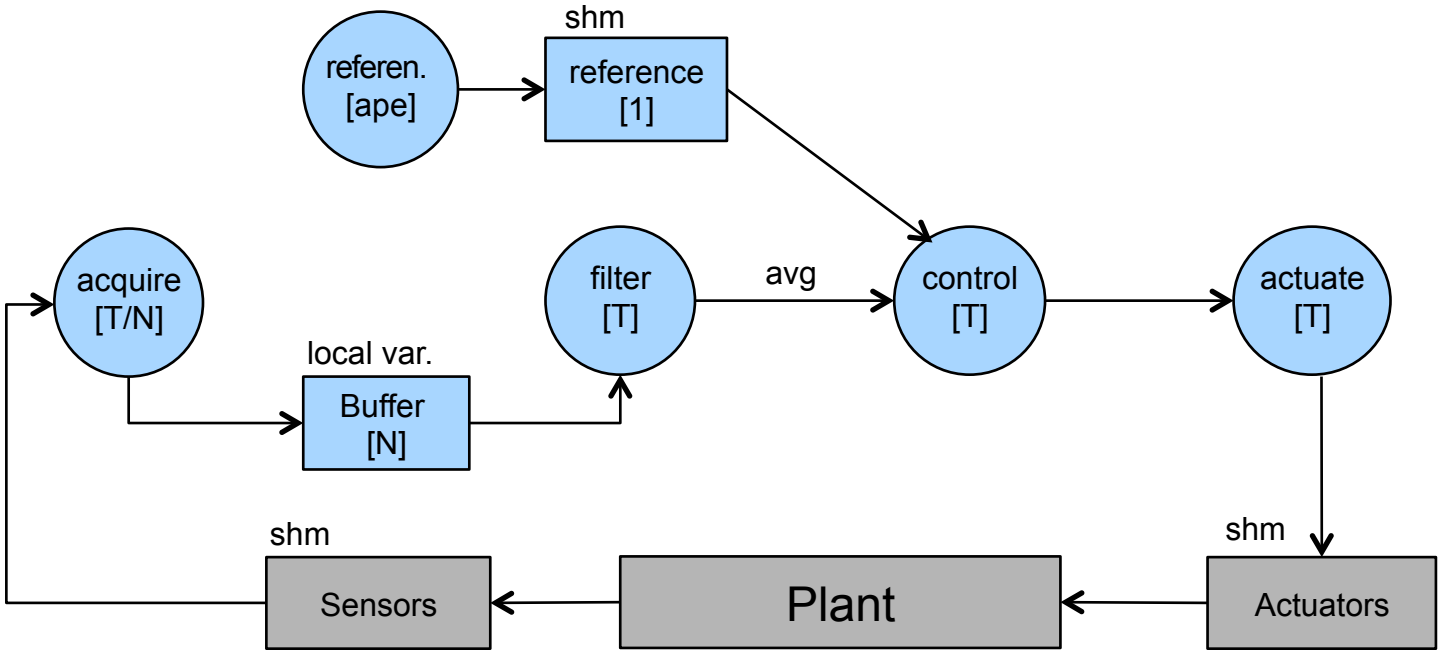
Corso di Sistemi Real Time

Marcello Cinque

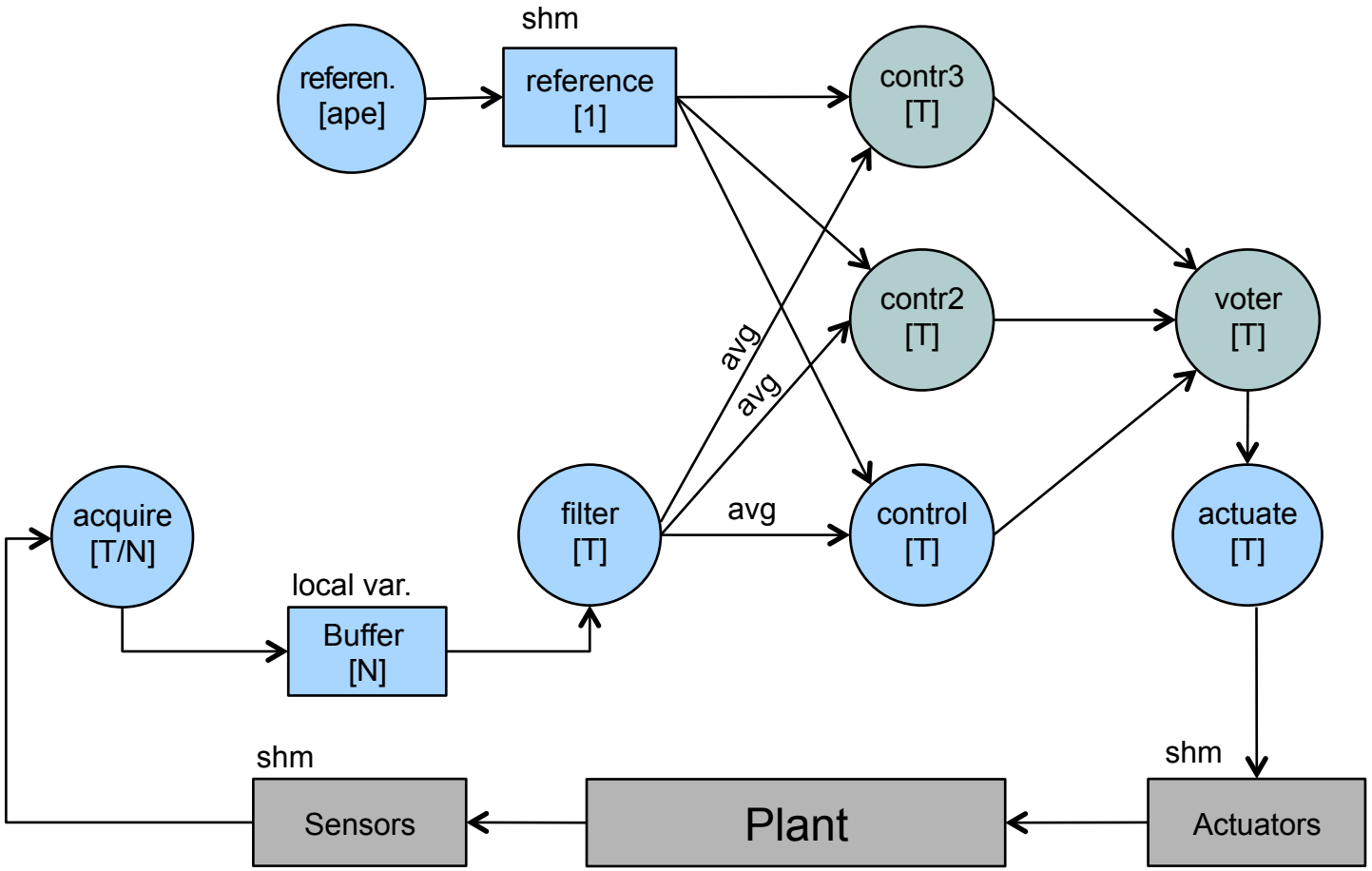
Traccia

- Si introduca un meccanismo di triple modular redundancy (TMR) per la tolleranza ai guasti nell'applicazione controller_ipc (pacchetto **controller_ipc.tar.gz** disponibile sul sito docente) al fine di aumentare l'affidabilità del task "control".
- Si introduca inoltre un meccanismo per il recupero dello stato dell'applicazione (diagnosi) attraverso un task asincrono gestito con un Deferrable Server
- Tali meccanismi devono essere realizzati attraverso task real time sfruttando le primitive IPC di RTAI

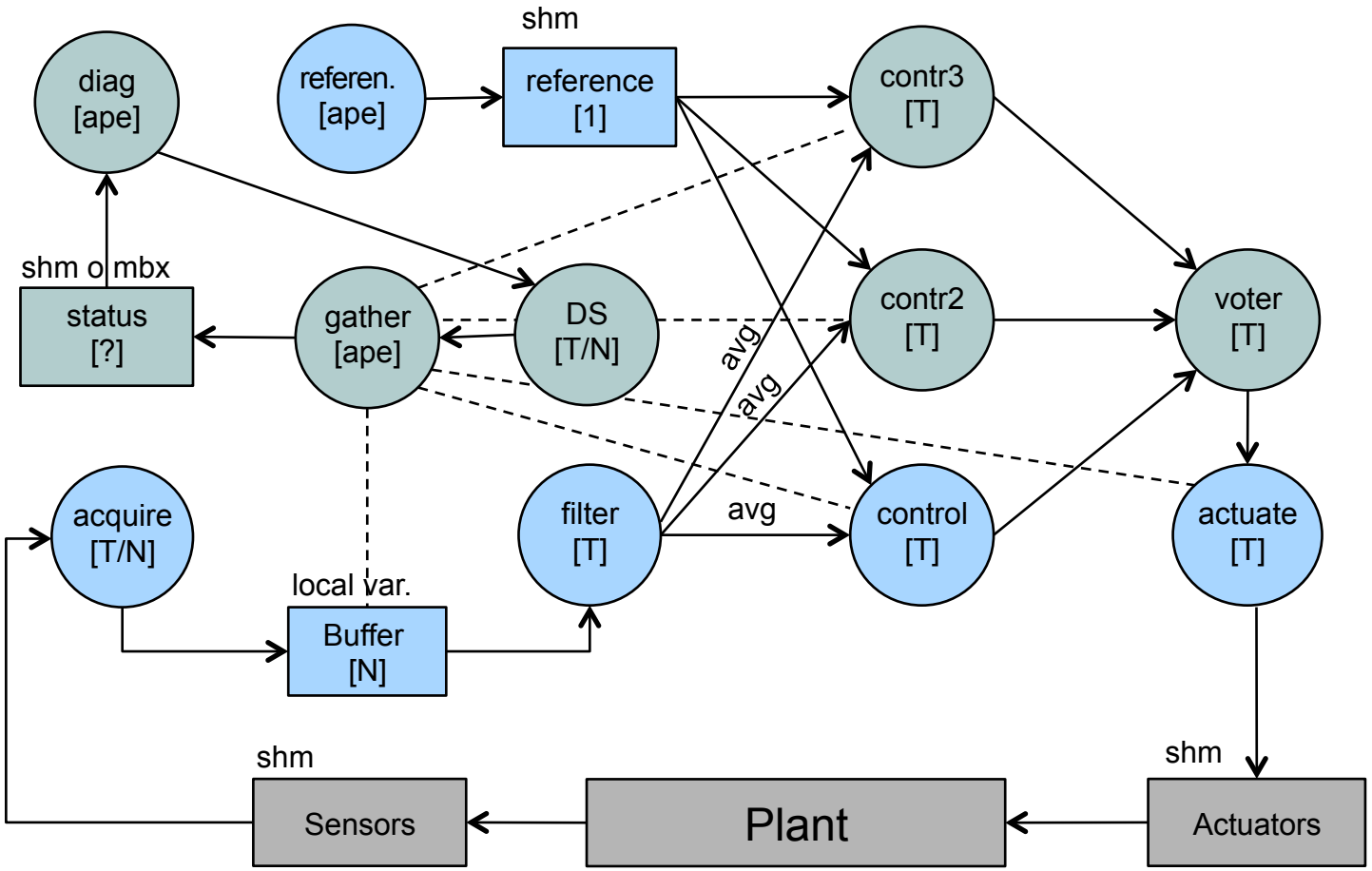
Schema di partenza



Schema da realizzare (1/2)



Schema da realizzare (2/2)



TMR

- I task ***control***, ***contr2*** e ***contr3***, più il task ***voter***, costituiscono lo schema TMR.
- I tre task di controllo ridondanti elaborano la stessa legge di controllo e la inviano al *voter*.
- Il *voter* raccoglie le tre decisioni, sceglie la decisione finale a maggioranza, e la invia al task *actuate*.
- Lo schema è in grado di tollerare:
  - crash (o ritardi eccessivi) di 2 repliche “control” su 3
  - errori di valore (ad es, legge di controllo errata) di 1 replica “control” su 3

Diagnosi con DS (1/2)

- I task ***diag***, ***gather*** e ***DS*** consentono la diagnosi dello stato del sistema in qualunque istante l'utente lo richieda.
- Il task *diag* è un task aperiodico non real-time realizzato in un programma a parte che, quando avviato, effettua una richiesta per il recupero dello stato dell'applicazione.
- La richiesta viene intercettata dal task *DS* che implementa un Deferrable Server:
  - È in attesa di richieste aperiodiche per tutto il suo periodo T/N
  - Appena riceve la richiesta, la serve immediatamente invocando il task aperiodico di servizio *gather*
  - Per semplicità, si ometta il controllo della capacità del server, facendo l'assunzione che il task *gather* abbia un WCET inferiore della massima capacità assegnabile al DS

Diagnosi con DS (2/2)

- Quando invocato, il task *gather* effettua il recupero del seguente stato del sistema:
  - il contenuto corrente del Buffer
  - i valori correnti dei segnali di controllo elaborati dai tre task di controllo
  - gli stati dei tre task di controllo (attivo/fallito)
  - il valore del segnale di controllo deciso dal voter
- avendo cura di
  - non leggere stato inconsistente dal Buffer (ad es., se il Buffer è correntemente letto da *gather*, non deve essere scritto da *aqcuire* o letto da *filter*)
  - non interferire con i tre task di controllo e il *voter*, evitando che questi debbano bloccarsi o sincronizzarsi per inviare il proprio stato a *gather*
- Una volta recuperato, lo stato viene salvato in una memoria condivisa (o inviato tramite mailbox) e riportato a video dal task *diag*

Testing

- Si testi il funzionamento del TMR e della diagnosi “iniettando” dei fault nei task di controllo
  - introducendo delle “sleep” o delle wait_until nell’ordine delle decine di secondi in uno/due task di controllo
  - terminando in anticipo uno/due task di controllo
  - simulando un errore del tipo “stuck-at-N” in un task di controllo, ad es., segnale di controllo bloccato sempre allo stesso valore
- e verificando che il meccanismo di ridondanza e voting funzioni, attraverso la diagnosi dello stato del sistema (dopo aver iniettato il fault) tramite invocazione del task *diag*.

Diversity (opzionale)

- Al fine di evitare **fallimenti di modo comune** dovuti agli stessi difetti nel software, le repliche *contr2* e *contr3* devono essere funzionalmente equivalenti al task primario *control*, ma implementate diversamente, ad esempio:
  - cambiando l'ordine delle istruzioni o facendo i calcoli in maniera diversa (*algorithmic diversity*)
  - realizzando le repliche in modo kernel (*technical diversity*)
- Si provi (opzionalmente) a realizzare le repliche *contr2* e *contr3* adottando uno schema di diversity

Schedulabilità (opzionale)

- Si provi (opzionalmente) ad impostare un'analisi di schedulabilità del sistema
  - Misurando o assegnando in maniera fittizia i tempi C_i e le durate di eventuali sezioni critiche gestite con P.I. al fine di calcolare i tempi di bloccaggio B_i
  - Considerando la presenza del DS come task a priorità massima
  - Ignorando la presenza dei vincoli di precedenza tra i task dovuta allo scambio di messaggi