

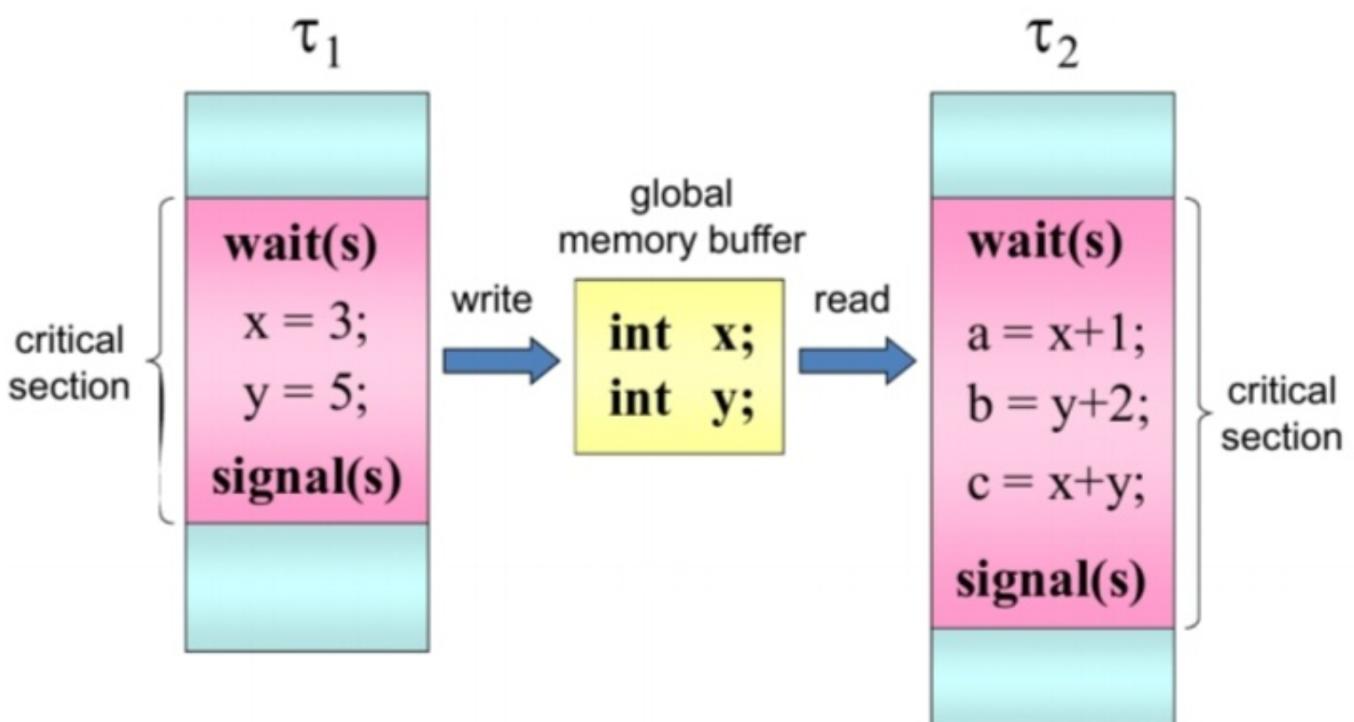
## DEFINIZIONE RISORSE

IL PROCESSORE È UN NODO  
DI RETE DI RISORSE

una risorsa è una risorsa per un utente software utilizzata da un task durante la sua esecuzione. si distinguono tra:

- **RISORSE PRIVATE**: risorse di un task specifico
- **RISORSE CONDIVISE**: condivise da task diversi
- **RISORSE ESCLUSIVE**: una risorsa condivisa che viene protetta contro l'accesso concorrente tramite l'uso di protocolli di accesso alle risorse i quali sono meccanismi che garantiscono il mutuo accesso alle risorse condivise.

Infine viene definita la **SEZIONE CRITICA** come la parte di codice che viene eseguita in mutua esclusione



## INVERSIONE DI PRIORITÀ

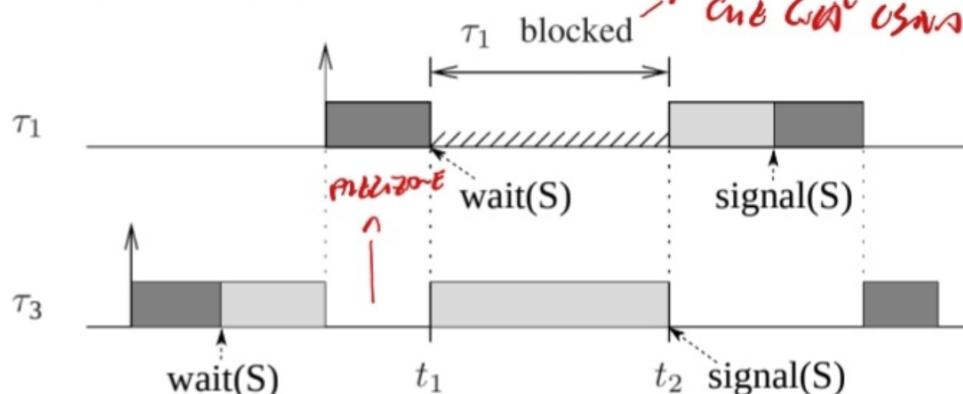
L'inversione di priorità è un fenomeno per il quale si ha che un task ad alta priorità blocca un task a bassa priorità per un intervallo di tempo rilevante nell'utente ma connessa connessa.

PUNTI SUL TASK  $T_1$  E  $T_3$  CHE  $P_1 > P_3$  e avviene una inversione causata da un semaforo binario  $S$ , sia che il tempo di blocco di  $T_1$  sia quello necessario a  $T_3$  per eseguire la sezione critica.

→ Poi allora non c'è inversione di priorità

■ normal execution

■ critical section

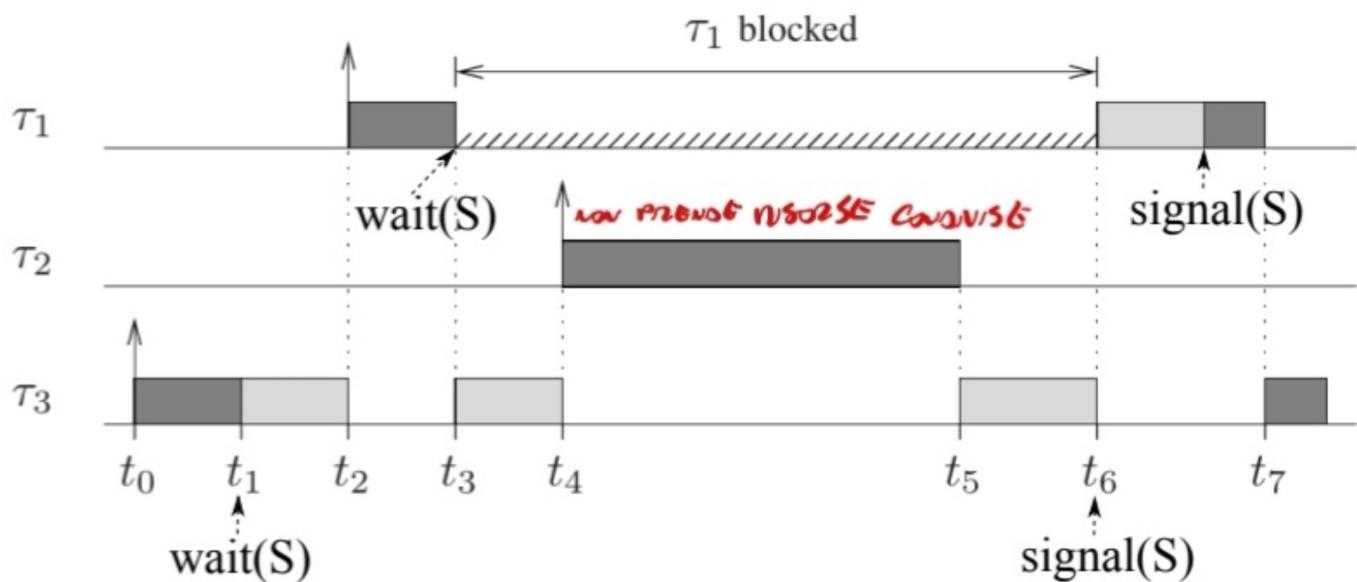


Adesso, se aggiungiamo un terzo task  $T_2$  ( $P_1 > P_2 > P_3$ ) si ha che il tempo massimo di blocco di  $T_1$  dipende non solo dalla durata della sezione critica di  $T_3$ , ma anche dal WGT di  $T_2$ .

||  
INVERSIONE DI  
PRIORITÀ!

normal execution

critical section



→ IL PROBLEMA SI RISOLVE UTILIZZANDO I PROTOCOLLI DI ACCESSO ALLE RISORSE!

QUESTIONI:

- UN INSIEME DI M TASSI PERIODICI  $T = \{T_1, \dots, T_m\}$  DOVE Ogni TASSI POSSIGGE UN:
  - PROTOKOLLO (SISTEMA)  $P_i$  CHE VENG ASSOCIAZIONE DEL SERVIZIO PER
  - PROTOKOLLO DI RISORSI  $P_i \geq P_i'$  CHE VENG NEL TEMPO E CHE VIENE ASSOCIAZIONE AL VALORE  $P_i$ .
- UN INSIEME m DI RISORSE CONDIVISE  $\Psi = \{R_1, \dots, R_m\}$  DOVE OGNI RISORSA VENG CESTITA SU UN SERVIZIO S

E LE SEGUENTI ASSUNZIONI SOVRASTANTI:

- Sono rilasciati all'arrivo
- Hanno differenti priorità nominali
- Presentati in ordine di priorità nominale ( $P_1 > \dots > P_n$ )
- Le loro sezioni critiche sono protette dal semaforo.

Abbiamo che l'obiettivo è quello di ottimizzare il massimo tempo di blocco. Bi che un task  $i$  può subire

per fare questo vorremo identificare 3 aspetti per ciascun protocollo:

- reco di accesso: devo controllare se il task  $i$  blocca il task  $j$  <sup>wait</sup>
- reco di processo: devo controllare all'interno della sezione critica
- reco di risoco: devo controllare le richieste dei task bloccati <sup>signal</sup>

## Priority Inheritance Protocol (PIP)

### reco di accesso

Un task  $i$  si blocca all'entrata di una sezione critica  $j$ , se la risorsa  $j$  è già posseduta da un task a più basso priorità  $j'$ .

I task bloccati sono sempre in base alla loro priorità dinamica.

## Regole di Precesso

All'interno di una sezione critica associata alla risorsa  $R_i$ , un task esegue con la massima priorità del task bloccato su  $R_i$ . Punto) Il task  $T_j$  eredita la priorità più alta fra tutti i task che blocca!

$$P_j := \max \left\{ P_j, \max \left\{ P_i \mid T_i \text{ è bloccato su } R_i \right\} \right\}$$

Una risorsa ha **Proprietà Transitiva**: se  $T_3$  blocca  $T_2$  e  $T_2$  blocca  $T_1$ , allora  $P_3 = P_1$

## Regole di Rilascio

Quando  $T_j$  esce dalla sezione critica associata alla risorsa  $R_i$  si ha che:

- il semaforo  $S_u$  viene rilasciato ( $\text{sign}(S_u)$ )
  - il task a priorità più alta che era bloccato su  $S_u$  è rivelato
  - se nessun altro task è bloccato da  $T_j$ , allora  $P_j = P_0$ , altrimenti  $T_j$  eredita la priorità più alta fra i vari task che blocca
- mostrare la sua priorità statica iniziale!

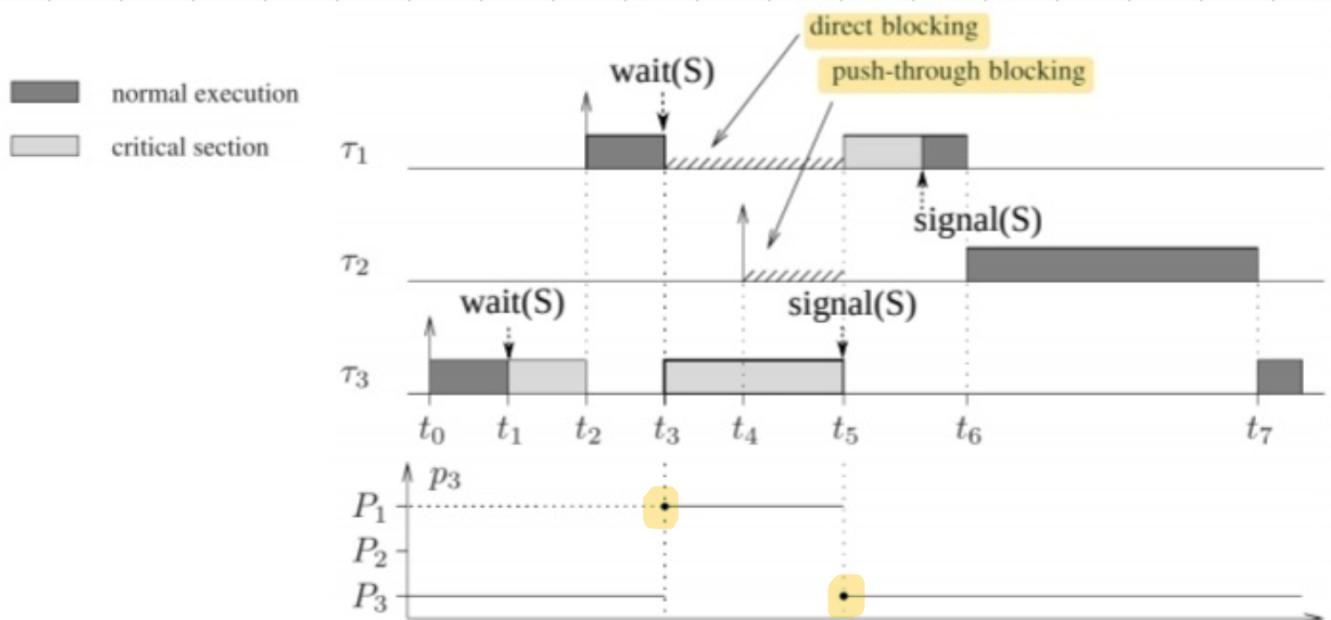
Esistono due diversi tipi di blocco:

(1)

**Blocco Diretto**: avviene quando un task ad alta priorità si

BLOCCA ALL'ENTRATA IN SEZIONE CRITICA DI UNA  
RISORSA CONDIVISA DA UN TASK A PIÙ BASSI PRIORITY  
E' NECESSARIO AL FINE DI GARANTIRE LA CONSISTENZA  
DELLA RISORSA CONDIVISA.

- ② **PUSH-THROUGH Blocking:** ANCHE quando un task A ha il priority  
VIGENTE blocca su un task A BASSO  
PRIORITY che ha però ordinato un  
Priority più alto da un altro task. E'  
NECESSARIO AL FINE DI EVITARE IL PENETRATO  
DI INVERSIONE DI PRIORITY!

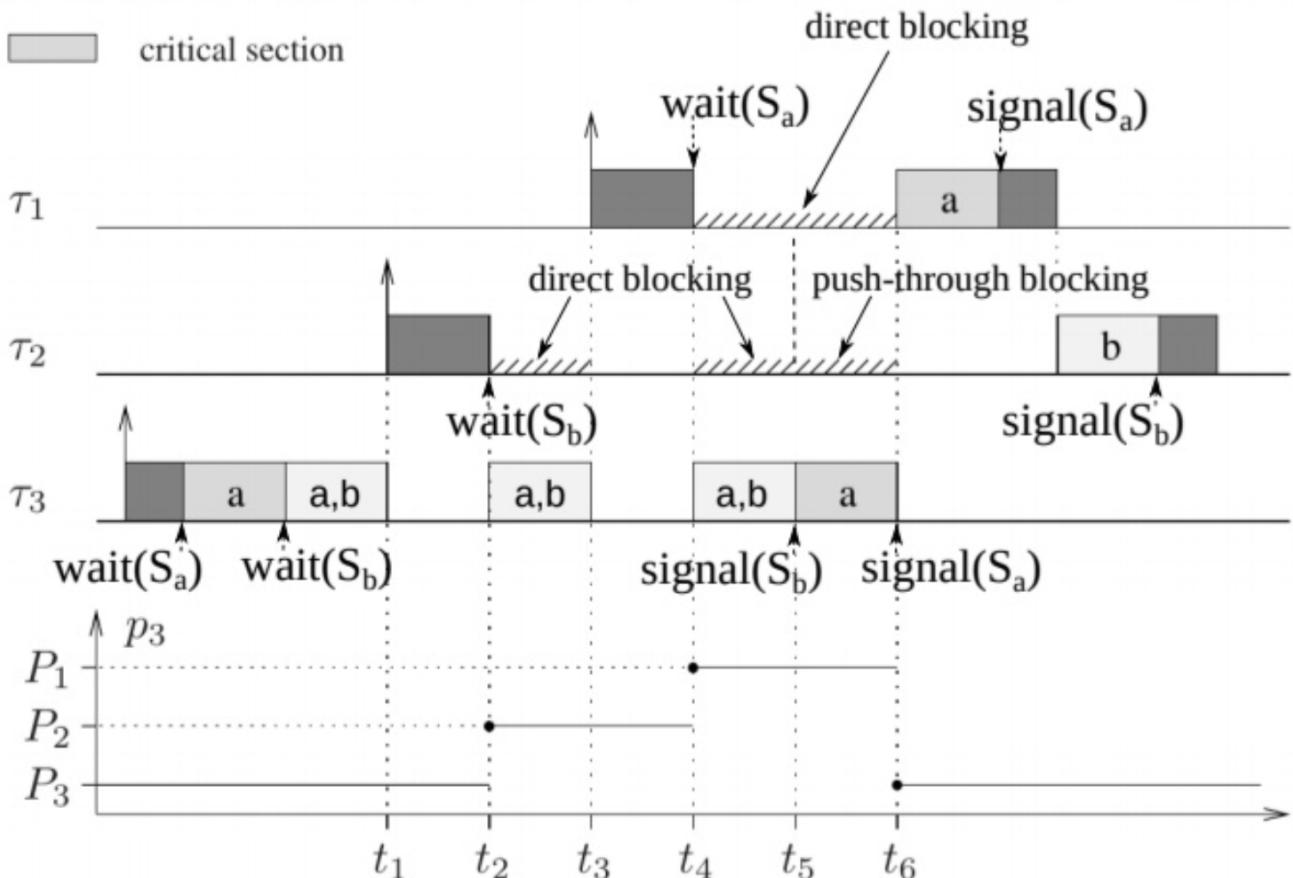


**ESEMPIO: SEZIONI CRITICHE ANNIDATE**

Si assume i task  $\tau_1, \tau_2, \tau_3$  dove  $\tau_1$  e  $\tau_3$  condividono la risorsa  
 No, mentre  $\tau_2$  e  $\tau_3$  condividono la risorsa  $R_b$ :

normal execution

critical section



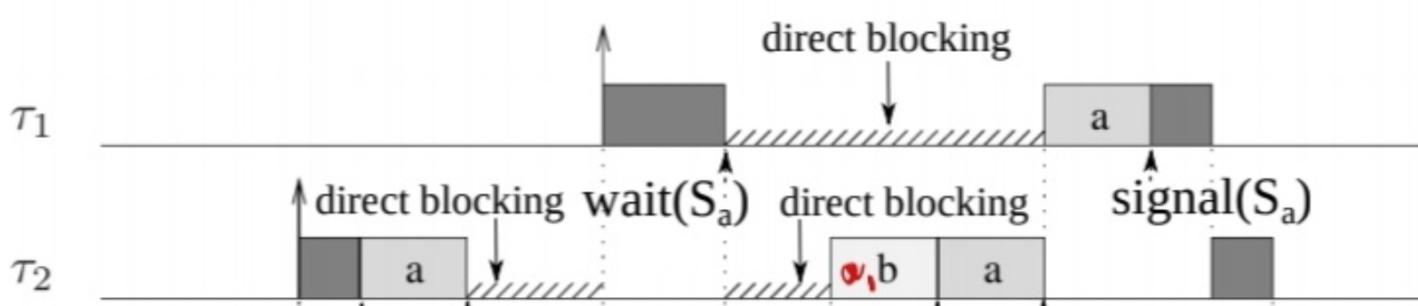
### ESEMPPIO: PROVVISORI TRANSITIVI

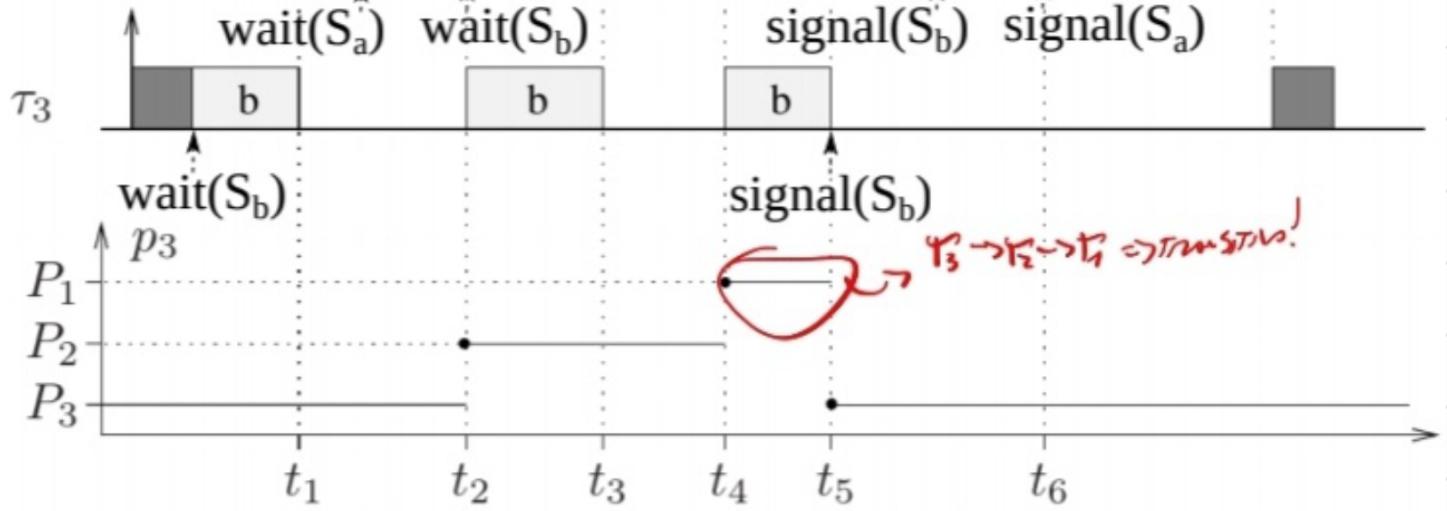
Si parla di PROVVISORI TRANSITIVI quando  $T_1$  è  $T_2$  condiviso da risorsa  $R$  mentre  $T_2$  è  $T_3$  condiviso da risorsa  $R$

All'istante  $t_4$  si ha che il task  $T_3$  eredita i provvisori del task  $T_1$  proprio il task  $T_2$  (transitivo)

normal execution

critical section





### PIP: PROPRIETÀ

- 1) UN SEMAFORO SU PROVVISORE BLOCCO PUSH-THROUGH AD UN TASSO  $T_i$  SOLO SE SU È ACCORDO DA UN TASSO A PROVVISORIO  $P_i$ . E UN TASSO A PROVVISORIO  $\geq P_i$
- 2) L'EREDITÀ DELLA PROVVISORIA TRANSITIVA SI HA SOLO QUANDO SI HANNO SISTEMI DI SEZIONI CRITICHE AVVOLGENTI
- 3) SE CI SONO  $l_i$  TASSI A PROVVISORIO BASSI CHE POSSONO BLOCCARE UN TASSO  $T_i$  ALLORA  $T_i$  PUÒ ESSERE BLOCCATO AL MASSIMO PER LA DURATA DI  $l_i$ . SEZIONI CRITICHE, OVEVA UNA PER OGNI TASSO A BASSA PROVVISORIA, A PRESOVOGLIO DAL NUMERO DI SEMAFORI USATI DAL TASSO  $T_i$ .

SE SONO PRESENTI SI SEMPRE DISTINTI CHE POSSONO BLOCCARE UN TASSI T<sub>i</sub> ANCHE T<sub>i</sub> PUÒ ESSERE BLOCCATO AL MASSIMO PER LA DURATA DI SI: SEZIONI CRITICHE, ANCHE LUNGA PER CIASCUA SEMIFORO A PRESCEDERE DAL NUMERO DI SEZIONI CRITICHE.

5)

TRAMITE IL PROTOCOLLO PIP, UN TASSI T<sub>i</sub> PUÒ ESSERE BLOCCATO AL MASSIMO PER LA DURATA DI:  $\alpha_i = \min\{l_i, s_i\}$  SEZIONI CRITICHE, DOVE:

- $l_i$  È IL NUMERO DI TASSI A PRIORI PIÙ BASSI CHE POSSONO BLOCCARE T<sub>i</sub>.
- $s_i$  È IL NUMERO DI SEMIFORI CHE POSSONO BLOCCARE T<sub>i</sub>.

### PIP: VANTAGGI E SVANTAGGI

#### VANTAGGI:

- UN TASSI VIENE BLOCCATO SOLAMENTE FINO A CHE È NECESSARIO.
- TRANSPARENZA AL PROGRAMMATORE.
- I TEMPI DI BLOCCO SONO LIMITATI AL MASSIMO DELLA DURATA DI OGNI SEZIONE CRITICA.

#### SVANTAGGI:

- COMPUTAZIONE DEI TEMPI DI BLOCCO COMPLESSI DONDE ALLA PRESENZA DEI VARI TIPI DI BLOCCO POSSIBILI (DIRECT, PUSA-THROUGH, TRANSITIVE).
- COMPLESSA IMPLEMENTAZIONE.

- Propenso al CHANNEL BLOCKING, ovvero quando CIASCUA TASK T<sub>i</sub> È BLOCCATA da TI; VOGLIATE NEL CASO PEGGIORI
- non PREVENE I DEADLOCKS causati dall'uso errato dei SEMFONI
  - ↳ quando ciascun processo è in attesa di un certo evento che solo un altro processo può produrre

## PRIORITY CEILING Protocol (PCP)

DEFINIAMO PRIMA LA PRIORITY CEILING C(S<sub>u</sub>) DI UN SEMFORO S<sub>u</sub> CONTE LA PROSSIMA PRIORITÀ TUTTI I TASK CHE POSSANO OTTENERE S<sub>u</sub>.

$$C(S_u) := \max_{i \in \{1, \dots, m\}} \{ P_i \mid T_i \text{ utilizza } S_u \}$$

### REGOLE DI ACCESSO

UN TASK T<sub>i</sub> SI BLOCCA ALL'ENTRATA DI UNA SEZIONE CRITICA SE LA SUA PRIORITÀ NON È PIÙ ALTA DEL CEILING MASSIMO DEI SEMFORI BLOCCATI DA ALTRE TASK.

$$\Rightarrow P_i \leq \max \{ C(S_u) \mid S_u \text{ blocca dal task } + T_i \}$$

↳ TEST DI ACCESSO!

QUINDI UN TASK NON PUÒ ENTRARE IN SEZIONE CRITICA UNTILLO DI UN SEMFORO LIBERO SE SONO PRESENTI SEMFORI CHE LO POSSANO BLOCCARE. QUINDI UNA VOLTA CHE UN TASK ENTRA NELLA SUA PRIMA SEZIONE CRITICA, NON PUÒ MAI ESSERE BLOCCATO DA UN TASK A PIÙ

BASSA PRIORITÀ FINO AL SUO COMPLETAMENTO.

- Record di Progresso
- Record di Accesso

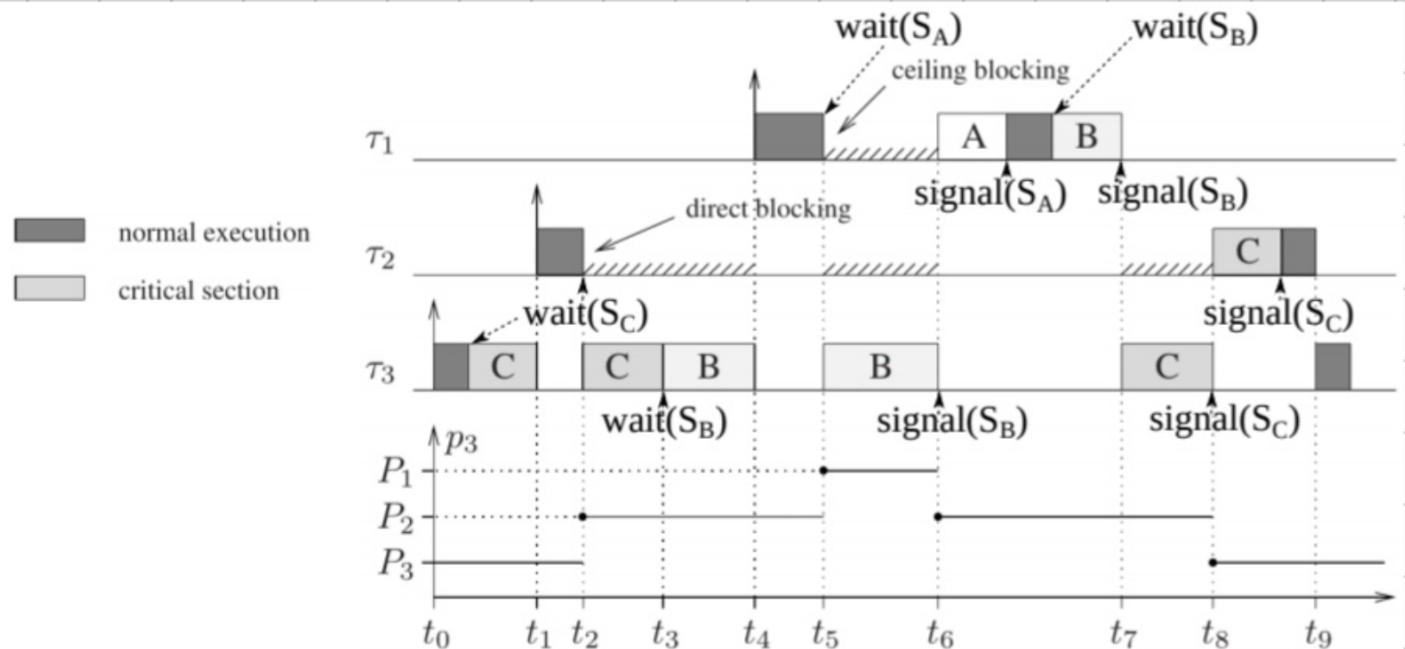
} IDENTICHE A quelle definite nel  
Protocols PIP!

## PCP: CEILING BLOCKING

AVVIENE quando una TASK È BLOCKED perché non ha SUFFICIENZA

### IL TGSJ DI ACCESSO DI PCP?

SI HANNO 3 TASK DOVE  $T_1$  USA UN RISORSO  $R_A \in R_B$ ,  $T_2$  USA  $R_C \in R_B$ ,  $T_3$  USA  $R_B, R_C$ :



## PROPRIETÀ PCP

1) SE UN TAU Y<sub>i</sub> È PRESENTE ALL'INTERNO DELLA SUA SEZIONE CRITICA DI UN TAU Y<sub>j</sub> CHE ENTRA IN SEZIONE CRITICA Z<sub>i,b</sub> ALLORA Y<sub>i</sub> NON PUÒ ESEGUIRE UNA PROVVISI → DELLA PROVVISI DI Y<sub>j</sub> FINO A QUANDO Y<sub>j</sub> NON È COMPLETATO

2) IL PRIORITY CELLS PROTOCOL PREVENE IL BLOCCO TRANSITIVO

3) IL PRIORITY CELLS PROTOCOL PREVENE I DEADLOCKS

4) SOTTO IL PRIORITY CELLS PROTOCOL UN TAU PUÒ ESSERE BLOCCATO PER AL MASSIMO LA DURATA DI UNA SEZIONE CRITICA

PCP: VANTAGGI E SVANTAGGI

VANTAGGI:

- UNTA I BLOCCI ALLA DURATA DELLE SEZIONI CRITICHE
- PREVENE I DEADLOCKS E I BLOCCI TRANSITIVI

SVANTAGGI:

- COMPLESSO DA IMPLEMENTARE
- PIÙ CAUSSE BLOCCHI NON NECESSARI
- MA È TRANSPARENTE AL PROGRAMMATORE  $\Rightarrow$  CEGLI UNO DEI DUE GESSI E APPositamente DEFINITI!

## TEST DI SCHEDULABILITÀ ESTESI

NEL CASO IN CUI SI ABBIANO DEI TEMPI DI BLOCCO LIMITATI SI VUOLE AD ESTENDERE I TEST DI SCHEDULABILITÀ PER I TASK INDEPENDENTI. QUESTO CI PERMETTE DI GARANTIRE UN TASK ALLA VOLTA.

ESSENDO CHE LE CONDIZIONI DI BLOCCO DEVONTE NEL WORST CASE SCENARIOS DIFFERISCONO PER OgniCUN TASK E NON POSSONO PURO ACCADERE IN MODO SIMULTANEO, SI AVRA' CHE I TEST SONO SOLO SUFFICIENTI!

1)

### UTILIZZAZIONE BASED ANALYSIS:

- LL PER RM: UN INSERIMENTO DI TASKI PERIODICI CON FATTORI DI BLOCCO E CON  $D_i = T_i$  SI HA CHE OgniCUN TASK  $T_i$  È SCHEDULABILE DA RM SE:

$$\sum_{k | P_k > P_i} \frac{C_k + C_i + B_i}{T_k} \leq i(2^{n_i} - 1) + T_i$$

HR PER RM

$$\prod_{k|P_n > P_i} \left( \frac{C_k + 1}{T_k} \right) \left( \frac{C_i + B_i}{T_i} + 1 \right) \leq 2 \quad \forall T_i$$

• UC PER EDF

"

"

"

$$\sum_{k|P_n > P_i} \frac{C_k}{T_k} + \frac{C_i + B_i}{T_i} \leq 1 \quad \forall T_i$$

2)

## RESPONSE TIME ANALYSIS

UN INSISTEME DI TASSI PERIODICI CON FATTORI DI BLOCCO G  $D_i \leq T_i$   
È SCHEDULABILE DA DM SE:

$$R_i = C_i + B_i + \sum_{k|P_n > P_i} \left\lceil \frac{R_i}{T_k} \right\rceil C_k \leq D_i \quad \forall T_i$$

È UNA SOLUZIONE ITERATIVA PER DETERMINARE  $R_i$ :

$$R_i^{(0)} = \sum_{k|P_n > P_i} C_k + C_i + B_i$$

$$R_i^{(j)} = C_i + \beta_i + \sum_{k | P_k > P_i} \left[ \frac{R_i^{(j-1)}}{T_k} C_k \right]$$

3)

### PROCESSOR DEMAND ANALYSIS

UN INSERIMENTO DI TASK PERIODICO CON FATOR DI BLOCCHI  $D_i \leq T_i$   
 È SCHEDULABILE SE POSSO ESSERE:

$$U_{i,1} \in dbf(t) + \beta_i(t) \quad \forall t \in D$$

DOVE:

$$dbf(t) = \sum_i \left\lfloor \frac{t + T_i - D_i}{T_i} C_i \right\rfloor$$

$$\beta_i(t) = \max_{j, j \neq i} \{ \beta_{i,j} \mid D_i > t \wedge D_j \leq t \}$$

E' DEFINITA FLUSHING DI BLOCCHI,

ANCHE IL TEMPO MASSIMO PER IL QUALE  $T_i$  CON  $D_i \leq T_i$  PUÒ ESSERE  
 BLOCCATO DA  $T_j$  CON  $D_j > t$

$\beta_{i,j} :=$  TEMPO MASSIMO PER IL QUALE  $T_i$  MANTIENE UNA RISORSA A UN QUALE SISTEMA  
 ANCHE NELL'ULTIMA DELL'TASK  $T_j$

$$D := \{ d_i \mid d_i \leq \max \{ D_{\max}, \min \{ H, t \} \} \}, \quad D_{\max} := \max_i \{ D_i \}$$

$$t^* := \sum_i \frac{(r_i - d_i) u_i}{1 - U}$$