Tämä on esimerkki kilpailutilanteesta (race condition), joka voi syntyä, kun useat prosessit tai säikeet pyrkivät käyttämään jaettua resurssia, kuten kriittistä aluetta.

Tässä ongelmassa on seuraavat pääpiirteet:

Turnien jakaminen: Prosessit 0 ja 1 kilpailevat saadakseen vuoronsa (turn) suorittaa kriittisen alueen. Kun turn on annettu prosessille 1, se voi alkaa suorittaa kriittistä aluetta.

Prosessin 1 pitkäaikainen suoritus kriittisessä alueessa: Prosessi 1 saa vuoronsa suorittaa kriittisen alueen ja tekee sen onnistuneesti. Kuitenkin sen jälkeen, sen ei tarvitse antaa vuoroa takaisin prosessille 0, vaan se menee suorittamaan pitkäaikaista ei-kriittistä osuutta.

Prosessin 0 odottaminen: Prosessi 0 on jo antanut vuoron prosessille 1 ja odottaa, että se saa vuoronsa takaisin. Se ei kuitenkaan saa vuoroa takaisin pitkään aikaan, koska prosessi 1 ei ole valmis antamaan sitä takaisin.

Tämän seurauksena prosessi 0 jää odottamaan vuoroaan pitkään aikaa, eikä se pääse suorittamaan kriittistä aluettaan, vaikka se ei ehkä tarvitsisi odottaa niin kauan. Tämä on tehokkuusongelma, joka johtaa resurssin tehottomaan käyttöön ja voi hidastaa koko järjestelmän toimintaa.

Ratkaisuksi tähän ongelmaan voidaan harkita muita menetelmiä, kuten parempia aikataulutusalgoritmeja tai aktiivista odottamista (active waiting) käyttäviä ratkaisuja, jotka mahdollistavat prosessin 0 odottamisen vuoroaan tehokkaammin ilman, että se jää lukkiutumaan pitkäksi aikaa. Esimerkiksi voisi käyttää "spin lock" -lukitusta, jossa prosessi 0 odottaisi vuoroaan aktiivisesti, mutta voisi silti päästä suorittamaan kriittisen alueen, kun vuoro on vapaa, sen sijaan että se jäisi passiivisesti odottamaan. Tämä voisi parantaa resurssin käytön tehokkuutta.