Stany procesu Babci (B):

1. **WAIT** – Babcia odpoczywa i odpowiada komunikatem ACK na każde przychodzące REQ_JAR.

2. **WANT_JAR** – chce zdobyć słoik i wysyła REQ_JAR do wszystkich innych babć.

3. **MAKING_JAM** – Proces babci wchodzi do sekcji krytycznej w której posiada słoik i tworzy konfiturę.

4. **RELEASE_JAM** – Wychodzi z sekcji krytycznej po stworzeniu konfitury, zwalniając zasoby i wysyłając RELEASE(JAM) do wszystkich. U studentek komunikat zwiększa liczbę dostępnych konfitur K o 1 a u babć zmniejsza liczbę dostępnych słoików P o 1.

Stany procesu Studentki (S):

1. **WAIT** – Studentka nie ubiega się o konfitury i odpowiada komunikatem ACK na każde przychodzące REQ_JAM.

2. **WANT_JAM** – chce zdobyć konfiturę i wysyła REQ_JAM do wszystkich innych studentek.

3. **EATING_JAM** – Proces studentki wchodzi do sekcji krytycznej, gdzie studentka zjada konfiturę.

4. **RETURN_JAR** – wysyła RELEASE_JAR – wychodzi z sekcji krytycznej po zjedzeniu konfitury, zwalniając zasoby i wysyłając RELEASE_JAR do wszystkich. Oddaje babciom pusty słoik i zwiększa ich lokalną liczbę słoików P o 1. A studentką zmniejsza liczbę konfitur K o 1.

Opis działania algorytmu:

Na początku wszystkie procesy (zarówno babcie, jak i studentki) znajdują się w stanie WAIT i odpoczywają przez losowy czas. W tym stanie odpowiadają komunikatem ACK na każde przychodzące REQ, aktualizując swój zegar Lamporta. Po zakończeniu okresu oczekiwania:

- Babcia przechodzi do stanu WANT_JAR, wysyła REQ(JAR) do reszty babć i wstawia swoje żądanie do lokalnej kolejki priorytetowej.
- Studentka przechodzi do stanu WANT_JAM, wysyła REQ(JAM) do pozostałych studentek i również umieszcza swoje żądanie w lokalnej kolejce.

Każdy proces, który otrzyma REQ, aktualizuje swój lokalny zegar Lamporta, dodaje żądanie do swojej kolejki, a następnie odsyła ACK. Każde żądanie REQ posiada priorytet oparty na znaczniku czasowym Lamporta oraz ID procesu. Dzięki temu system działa sprawiedliwie i starsze żądania są realizowane wcześniej. System posiada dwie niezależne sekcje krytyczne:

- JAR – dotycząca dostępu do słoików, do której wchodzą procesy babć (ograniczenie: P)
- JAM – dotycząca dostępu do konfitur, do której wchodzą procesy studentek (ograniczenie: K)

Proces może wejść do sekcji krytycznej tylko wtedy, gdy:

1. Otrzyma ACK od wszystkich innych procesów swojej grupy.

2. Jego własne żądanie znajduje się na odpowiednim miejscu w lokalnej kolejce – tj.:

- W przypadku JAR: proces znajduje się wśród pierwszych P (czyli tylu, ilu mamy dostępnych słoików).
- W przypadku JAM: proces znajduje się wśród pierwszych K (czyli tylu, ile mamy dostępnych konfitur).

Po zakończeniu działania w sekcji krytycznej:

- Babcia przechodzi do stanu RELEASE_JAR, wysyła RELEASE(JAR) do wszystkich procesów i zwiększa K o 1.
- Studentka przechodzi do stanu RETURN_JAR, wysyła RELEASE(JAR) do wszystkich procesów i zwiększa P o 1.

W czasie przebywania procesu w sekcji krytycznej wszystkie przychodzące REQ_JAR są zapisywane do lokalnej kolejki i obsługiwane dopiero po zakończeniu działania. Wiadomość RELEASE powoduje, że inne procesy usuwają odpowiednie żądanie z lokalnej kolejki, a stan systemu się aktualizuje. Następnie proces wraca do stanu WAIT, czeka losową jednostkę czasu i cykl się powtarza.