Buffering

Quando noi andiamo a spedire un messaggio chiamiamo un servizio del sistema che eventualmente si fa carico di copiare le informazioni del messaggio, quindi quelle che noi abbiamo specificato all'interno dell'area di memoria che stiamo utilizzando, è possibile che il sistema ci ritorni il controllo - in particolare se la send è sincrona - avendo acquisito queste informazioni e avendole copiate/bufferizzate da qualche altra parte. Questo ci permette poi di sovrascrivere il contenuto dell'area che precedentemente ospitava l'informazione, senza che la copia originale che era stata acquisita e quindi era in qualche modo stata eseguita dal software del sistema, sia ROVINATA da queste operazioni di sovrascrittura che possiamo effettuare su quell'area di memoria specifica.
Questo si chiama "BUFFERING", QUINDI MANTENIMENTO DELLE COPIE DEI MESSAGGI DA QUALCHE PARTE E OVVIAMENTE IL BUFFERING È TALE PER CUI, SE NOI LO ABBIAMO, non è necessario che quando noi spediamo un messaggio, sia anche impostata una receive() per andare ad utilizzare questo messaggio

In memoria kernel
- non è necessario che sia impostata una receive() all'atto dell'arrivo del messaggio
- la capacità di bufferizzazione può essere nulla, limitata o illimitata
- in caso di capacita' nulla un solo messaggio alla volta può essere in transito (tipico del randez-vous) – adeguato in contesto di spedizione bloccante/sincrona
- è possibile definire un timeout per la bufferizzazione di un messaggio, allo scadere del quale il messaggio viene scartato

In memoria utente
- è necessario che sia impostata una receive() all'atto dell'arrivo del messaggio
- in caso la receive() non sia impostata, il messaggio in transito puo' venire perso (ad esempio per spedizioni non bloccanti sincrone)

Chiaramente se il buffering non c'è e vogliamo eseguire una send sincrona, non possiamo che seguire quanto segue:
"Noi abbiamo impacchettato il nostro messaggio nell'address space, chiamiamo su un thread l'operazione di spedizione, e se non c'è buffering a livello di sistema operativo - e quindi non fa copie di queste informazioni per mantenerle e poi consegnarle al chiamante di una ricezione - questo thread (se la send è sincrona) viene messo in stato di blocco.
Ma blocco di cosa? Rispetto al fatto che un altro thread possa chiamare una ricezione e quest'ultima andrà a prendere il contenuto del messaggio nell'address space e a copiarlo direttamente nell'address space che ha chiamato quest'ultimo thread. Quindi il S.O non ha fatto copie intermedie di questi messaggi, si copia il messaggio direttamente nell'address space senza avere copie "intermedie" di queste informazioni su strutture dati del nostro sistema operativo.".

Quindi non abbiamo la necessità di avere una receive impostata per eseguire realmente questa consegna perché la consegna avviene in maniera temporanea a livello di una struttura dati di BUFFERING, che il sistema operativo mantiene.
La capacità di bufferizzazione può essere nulla, quindi chiaramente le informazioni che noi andiamo a mantenere a livello sistema per quanto riguarda i messaggi può essere NULLA, quindi dobbiamo avere la RECEIVE() impostata, oppure possiamo avere capacità limitata e in un modello IDEALE illimitata.
In caso di capacità nulla un solo messaggio per volta può essere in transito da una specifica sorgente, questo è un classico per il RANDEZ-VOUS, è possibile che questa spedizione randez-vous sia tale per cui: "Tanto tu devi rimanere fermo fino a che qualcuno non riceve quel messaggio", ed è INUTILE fare una copia intermedia, appena qualcuno arriva e vuole il tuo messaggio io lo copio nell'area di memoria destinazione senza buffering intermedio e poi eventualmente risveglio il thread e lo riporto nuovamente eseguibile in CPU.
Il buffering può anche essere gestito in memoria utente, non gestito dal kernel. In questo caso dobbiamo avere delle receive impostate nel momento in cui vogliamo far colloquiare le applicazioni utilizzando questi messaggi perché in caso in cui una receive non sia impostata, è possibile che il messaggio possa essere perso.
Perché in fin dei conti io potrei avere uno scenario in cui faccio una spedizione, quindi chiamo un servizio di spedizione del KERNEL di un messaggio che ho impacchettato all'interno dell'area di cui COMUNICO IL POINTER, è possibile che se non ci sono delle receive attualmente già chiamate per effettuare la copia di questa informazione da qualche altra parte in qualche altro address space, e la send è non bloccante/asincrona, il controllo venga ritornato al thread che possa sovrascrivere l'informazione in quell'area iniziale di cui avevo sopra comunicato il pointer. Però questa informazione che prima c'era scritta (prima della sovrascrittura) non è stata mai consegnata, perché chiaramente non c'è stata neanche una bufferizzazione di questa a livello sistema.