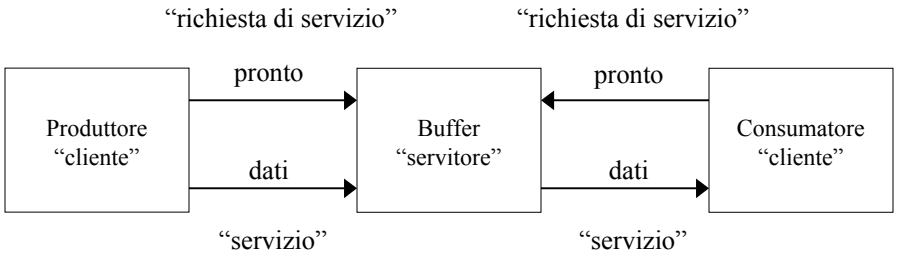
Primitive sincrone

- Processo mittente e processo ricevente *si sincronizzano* al momento della comunicazione.
- Il trasferimento dell'informazione avviene non appena entrambi i processi sono *pronti a comunicare* (*rendez-vous*).
- Il modello di comunicazione adottato prevede che ad un processo siano associati *canali privi di memoria*, uno per ogni tipo di messaggi che il processo può ricevere.

Produttore-consumatore

- Il processo buffer può contenere fino ad N messaggi.
- Utilizzo delle primitive.

  Send (*mess, proc*);
  Proc:= **Receive** (*mess*);
- Il processo produttore invia due tipi di messaggi, *pronto (*per tenere conto della limitazione di *buffer)* e *dati*.Il processo consumatore invia un messaggio *pronto.*
- Il processo *buffer* ha *due canali*, uno per il tipo di messaggio *pronto* ed uno per il tipo di messaggio *dati.*Essendo la **send** di tipo *sincrono* non è necessario, rispetto al caso di **send** *asincrona*, il messaggio *ok-to-send.*
- I canali servono al processo *buffer* per sincronizzarsi con i processi produttore e consumatore.



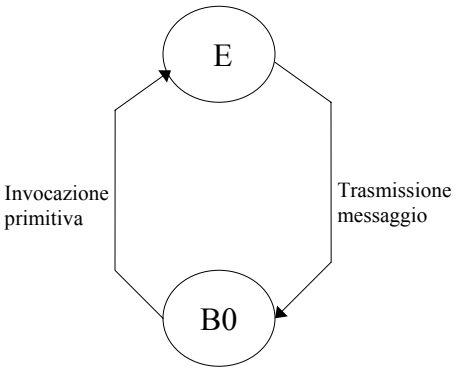

```
Process buffer;
var coda :<coda di elementi di tipo T>,
     proc:process;
     cons-pronto, prod-pronto :boolean initial false
     dati:T;
     pronto:signal
begin
   while true do
       begin
        proc:= Receive(pronto);
        case proc of
            produttore: if <coda piena>
                        then prod-pronto:=true;
                        else begin
                           proc:=Receive (dati);
                           if cons-pronto then begin
                                        Send (dati,consumatore);
                                        cons-pronto:=false;
                                             end;
                           else<inserzione dati in coda>;
                              end;
```

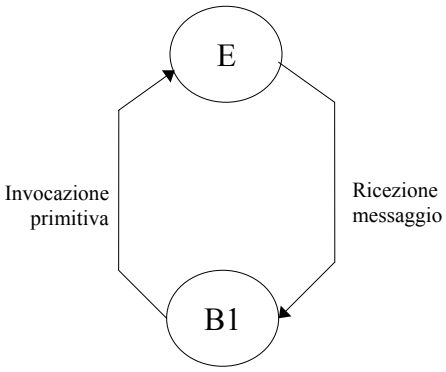
```
        consumatore: if <coda vuota>
                        then cons-pronto:=true;
                        else begin
                        <estrazione dati da coda>;
                        Send (dati , consumatore);
                        if prod-pronto then begin
                                        proc:= Receive(dati);
                                        <inserzione dati in coda>;
                                         prod-pronto:=false;
                                             end;
                                end;
        end;
end.
```

```
Process produttore;
  var dati:T; pronto:signal;
  begin
    while true do begin
                        <produci dati>;
                        Send(pronto,buffer);
                        Send(dati, buffer);
                        end;
  end;

Process consumatore;
  var dati:T; pronto:signal; proc:process
  begin
    while true do begin
                        Send(pronto,buffer);
                        proc:= Receive(dati)
                        <consuma dati>;
                        end;
   end;
```

• Stati di un processo che esegue la send e la receive:



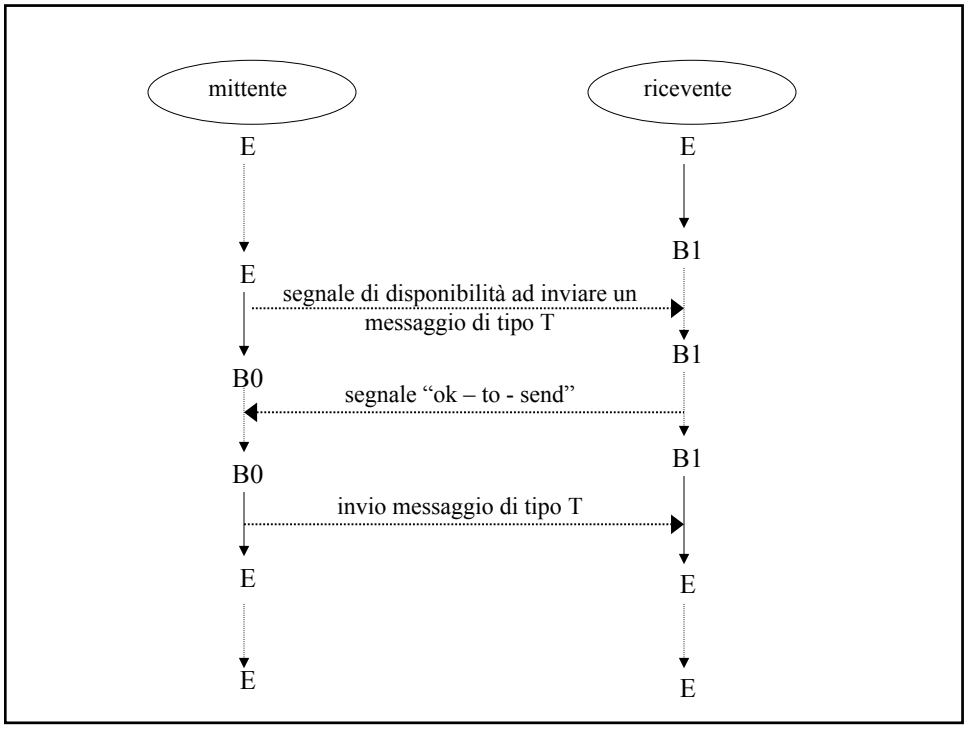


E (esecuzione)
BO (bloccato, ma pronto ad inviare un messaggio)

E (esecuzione)
B1 (bloccato, ma pronto a ricevere un messaggio)

- Il *supporto a tempo di esecuzione* ha il compito di implementare i *messaggi di sincronizzazione* necessari per garantire il comportamento sincrono delle primitive.
- La **Send** viene tradotta nella seguente sequenza di operazioni:

1) invio di un segnale di disponibilità a spedire un messaggio del tipo specificato;
2) attesa del segnale di disponibilità (*ok-to-send*) del ricevente (*sincronizzazione*);
3) invio del messaggio vero e proprio.


mittente
ricevente
E
E
B1
E
segnale di disponibilità ad inviare un
messaggio di tipo T
B1
B0
segnale “ok – to - send”
B1
B0
invio messaggio di tipo T
E
E
E
E