MPI-Message Passing Interface

An approach for Parallel Algorithm

UNDER GUIDANCE OF

PROF. UTPAL RAY

DEPARTMENT OF INFORMATION TECHNOLOGY . J.U.

About Us

- Niraj Kr Dubey
  - **001411001069**
- Subhrodip Mazumdar
  - **001411001007**

Goal

“To verify Chandy Lamport’s Global Snapshot Algorithm on a Distributed Money Transfer System “.

Project Outline

- First, implement money transfer system(lets say bank) which acts as one branch of the distributed system.
- These bank branches send money to each other at unpredictable time.
- We create different instance of this application on different host to form distributed system.
- Periodically or on user request one of the bank branch will initiate Chandy Lamport Global Snapshot Algorithm.

Background

- Programming Language.

  Base of Project.

- **C** for implementing Algorithm.
- **MPI** for Message Passing & Parallel Processing.

Background

- MPI

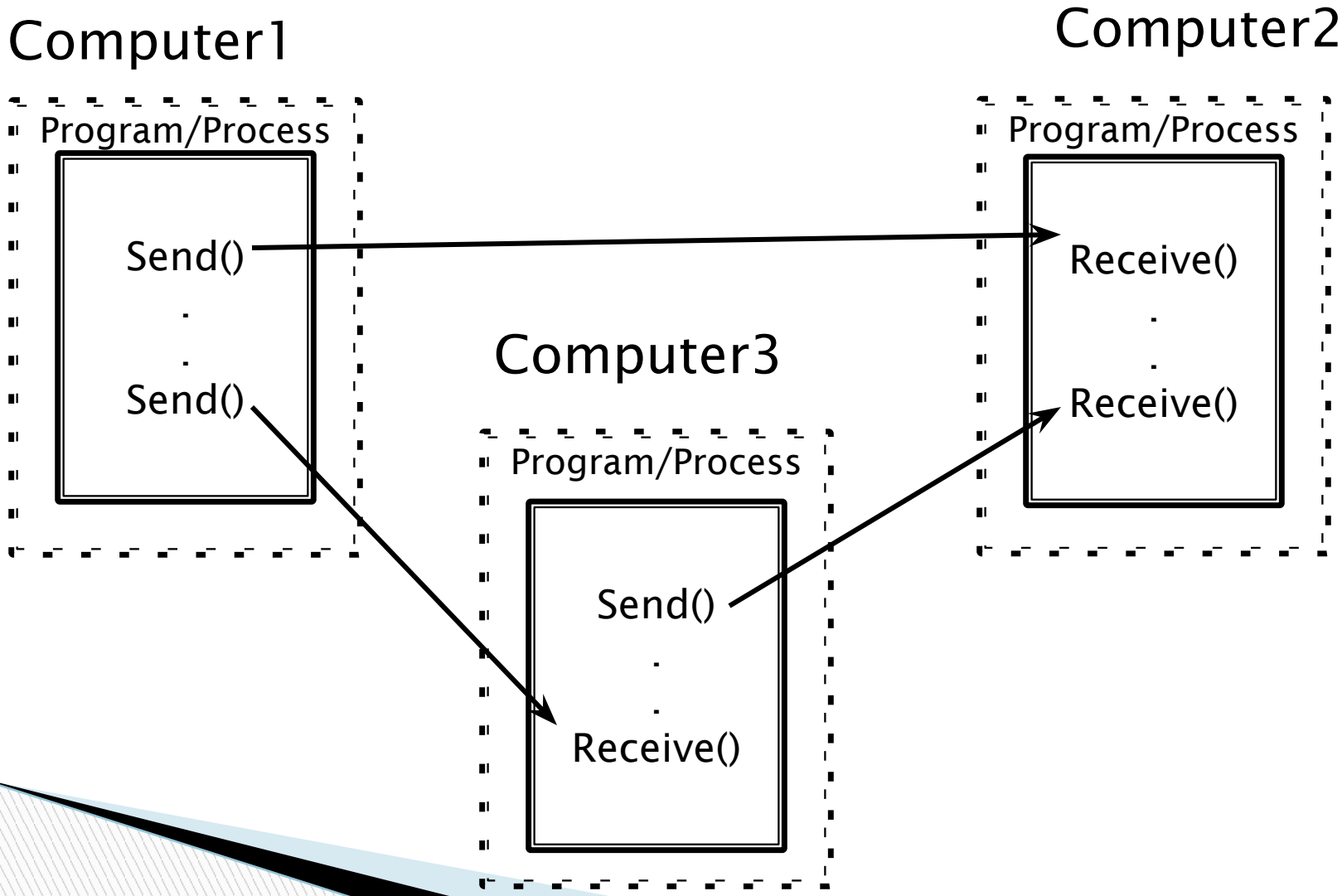
Message Passing Interface

- MPI is a proposed standard message -passing interface. It is a library specification not a language.
- The program written by the user are compiled ordinarily and linked with the MPI library.

MPI

- Message Passing library specification
  - Extended message-passing model.
  - Not a language or compiler specification.
  - Not a specific implementation, several implementations (like pthread).
- Standard for distributed memory, message passing ,parallel computing.
- Distributed Memory-Shared nothing approach.

Message passing concept using library routine.



Implementation.

- **Money Transfer System**- continuous executing system , transferring money from one node to another.

- **Chandy Lamport Algorithm**- which when initiated by any node, records the global state of the system and checks for consistency.

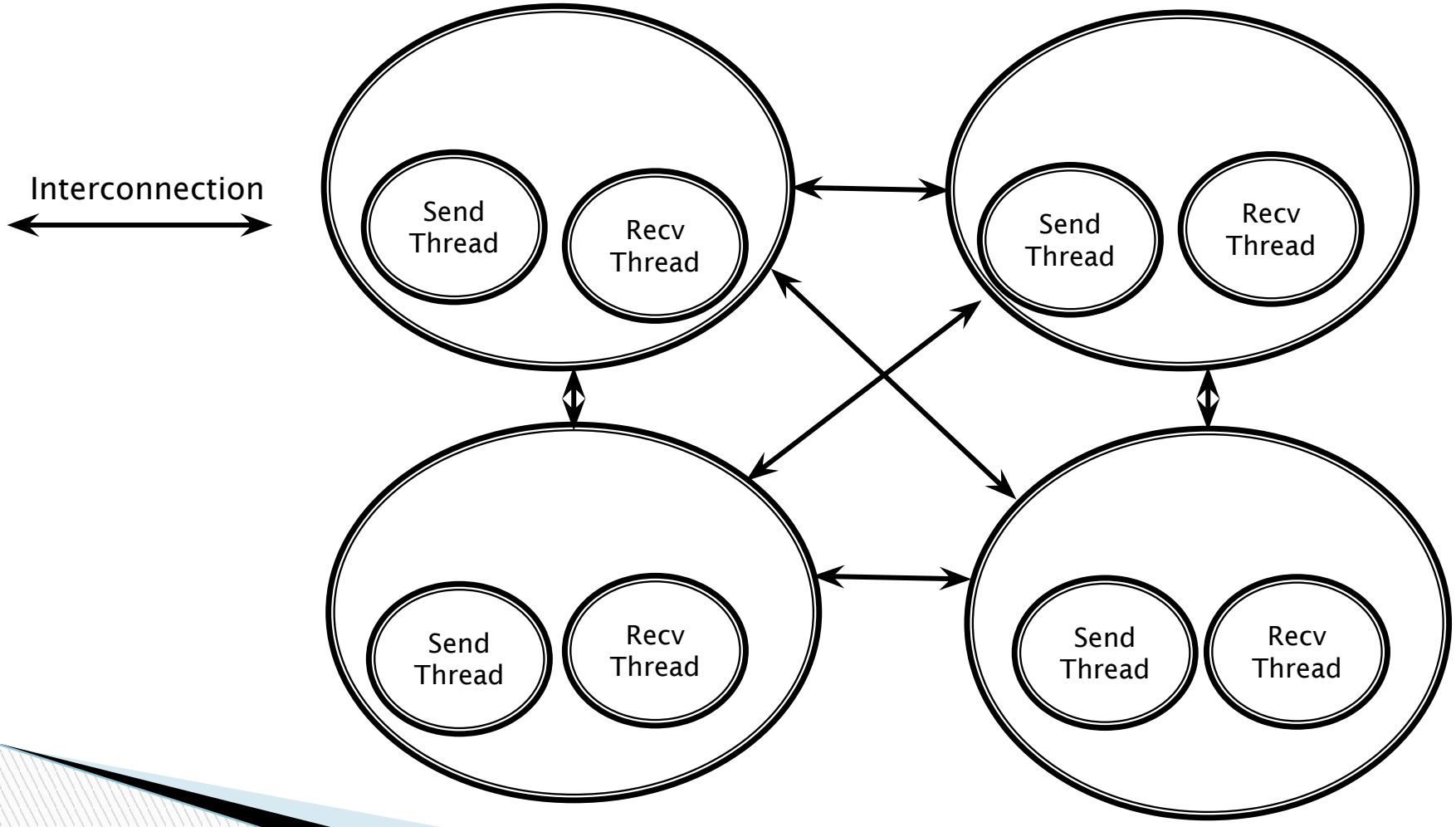
Money Transfer System

- The concept :
  - Each node will have a send and receive thread and will initially start with some fixed amount of money (Rs. 10000).
  - The **Send thread** will send a random amount of money to any arbitrary node.
  - The **Receive thread** will receive and update the node's money.

It will be a continuous process and will mimic the natural banking system.

Money Transfer System (Contd.)

- The architecture of the system :



Money Transfer System

- Importance of having two different threads :
  - Ensures the process of sending money is not dependent on receiving money and vice-versa.
  - If same thread is used, then independent behavior and concurrency can not be achieved simultaneously.
  - Reduces the load on the channel for message passing.

Global Snapshot Implementation.

- sendMarker():

Marker Sending Rule for process *i:*

begin

(i) Process ***i*** records its state.

(ii) For each outgoing channel C on which a marker has not been sent, ***i*** sends a marker along C before ***i*** sends further messages along C.

end

- recvMarker():

Marker Receiving Rule for process *j:*

On receiving a marker along channel C:

If *j* has not recorded its state **then**

begin

(i)Record the state of C as the empty set.

(ii)Follow the '**Marker Sending Rule'**.

end

else

Record the state of C as the set of messages received along C after ***j's*** state was recorded and before ***j*** received the marker along C.

- terminateSnapshot():
  - All processes have received a marker (and recorded their own state)
  - All processes have received a marker on all the N-1 incoming channels (and recorded their states)
  - Later, a central server can gather the partial state to build a global snapshot

Observation

```
Enter Initiator Process
INITIATOR:2
-------------------------------------------------------
   CHANDY        LAMPORTS          GLOBAL        SNAPSHOT
-------------------------------------------------------
   PROCESSOR    STATE     P0     P1     P2     P3
       0        9901      0      154    10     84
       1        10940     59     0      109    468
       2        8688      228    15     0      384
       3        7849      165    72     874    0

   TOTAL SUM=40000
```

Challenges

- The send and receive buffer should be taken care properly else distributed debugging will be a big problem .
- For a large number of MPI_Send and MPI_Recv operations Open MPI: version 3.0.0 must be used.
- Once the state of nodes has been recorded, no other message should be sent before the marker message.

Conclusion

- The system was able to fulfill both safety and liveliness property.
- The system state at any instant of time was consistent.
- Total Sum of money was always consistent.
- Further other variants of the Chandy Lamport's snapshot algorithm like **Spezialetti** , **Venkatesan's** and **Kearns** algorithm could be impelemented on this system.

THANK YOU..