



Leader Election in Distributed Systems

Course: Distributed Computing

Faculty: Dr. Rajendra Prasath

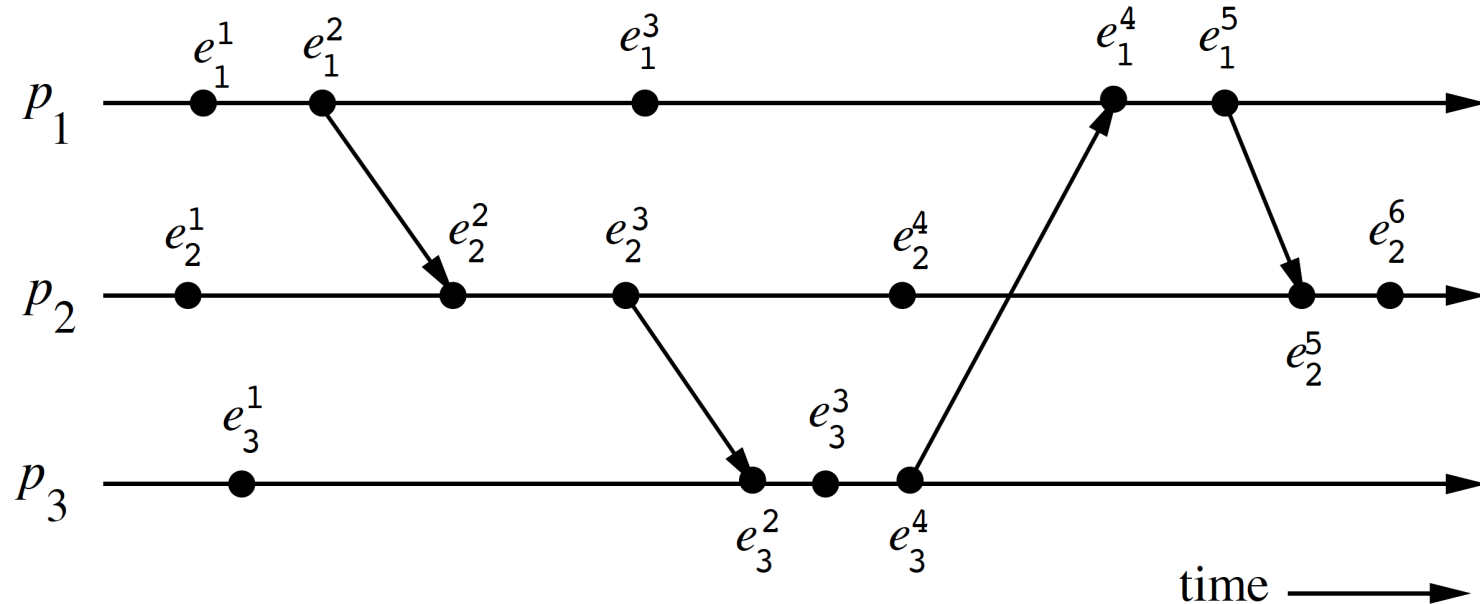
About this topic

This course covers the essential aspects of **Leader Election in Distributed Systems and its related concepts**

What did you learn so far?

- Challenges in Message Passing systems
- Distributed Sorting
- Space-Time Diagram
- Partial Ordering / Total Ordering
- Causal Ordering
- Causal Precedence Relation
 - Happens Before
- Concurrent Events
- Local Clocks and Vector Clocks
- Distributed Snapshots
- Termination Detection using Dist. Snapshots

A State-Time diagram - An Example



- For Process P'_1 :
- Second event is a message send event
 - First and Third events are internal events
 - Fourth event is a message receive event

Topics to focus on ...

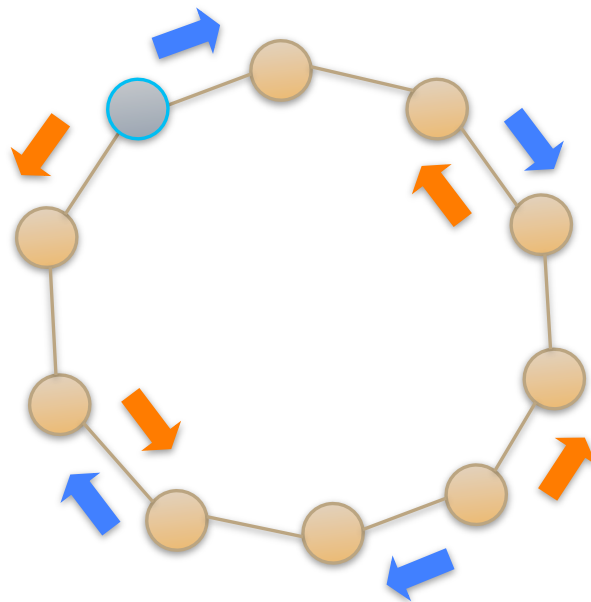
- **Leader Election in Distributed Systems**
- **Topology Abstraction and Overlays**
- **Message Ordering**
- **Group Communication**
- **Distributed Mutual Exclusion**
- **Deadlock Detection**
- **Check pointing and rollback recovery**

For Mid Semester 2

Leader Election in Distributed Systems

Ring Networks

- In an oriented ring, processes have a consistent notion of left and right



↘ Clockwise (right)
↗ Anti-clockwise (left)

- For example, if messages are forwarded on right channel, they will cycle clockwise around the ring

Why Study Rings?

- Simple starting point, easy to analyze
- Abstraction of a token ring
- Lower bounds and impossibility results for ring topology also apply to arbitrary topologies

Leader Election - Definition

- Each processor has a set of **elected** (won) and **not-elected** (lost) states.
- Once an elected state is entered, processor is always in an elected state (and similarly for not-elected): i.e., irreversible decision
- In every admissible execution:
 - every processor eventually enters either an elected or a not-elected state
 - exactly one processor (the **leader**) enters an elected state

Uses of Leader Election

- A leader can be used to coordinate activities of the system:
 - find a spanning tree using the leader as the root
 - reconstruct a lost token in a token-ring network
- We will study leader election in rings.

Anonymous Rings

- How to model situation when processes do not have unique identifiers?
- First attempt: Does each process require to be in the same state machine?
- Subtle point: Does the algorithm rely on knowing the ring size (number of processes)?

Uniform (Anonymous) Algorithms

- A **uniform** algorithm does not use the ring size (same algorithm for each size ring)
 - Formally, every processor in every size ring is modeled with the same state machine
- A **non-uniform** algorithm uses the ring size (different algorithm for each size ring)
 - Formally, for each value of n , every processor in a ring of size n is modeled with the same state machine A_n .
- Note the lack of unique ids.

Leader Election in Anonymous Rings

- **Theorem:** There is *no* leader election algorithm for anonymous rings, even if
 - algorithm knows the ring size (non-uniform)
 - synchronous model
- **Proof Sketch:**
 - Every processor begins in same state with same outgoing messages (since anonymous)
 - Every processor receives same messages, does same state transition, and sends same messages in round 1
 - Do the same for rounds 2, 3, ...
 - Eventually some processor is supposed to enter an elected state.

Leader Election in Anonymous Rings

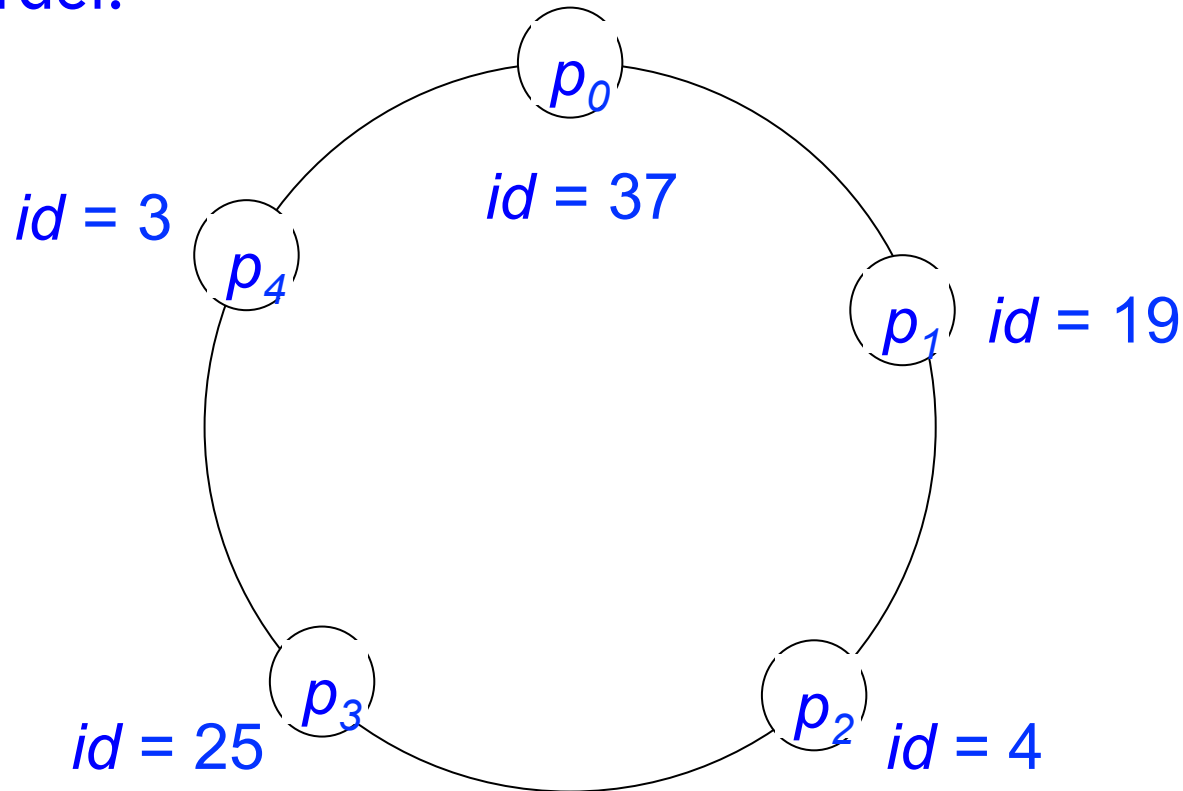
- Proof sketch shows that either **safety** (never elect more than one leader) or **liveness** (eventually elect at least one leader) is violated
- Since the theorem was proved for non-uniform and synchronous rings, the same result holds for weaker (less well-behaved) models:
 - uniform
 - asynchronous

Rings with Identifiers

- Assume each processor has a unique ID.
- Don't confuse indices and IDs:
 - **indices** are 0 to $n - 1$; used only for analysis, not available to the processors
 - **IDs** are arbitrary nonnegative integers; are available to the processors through local variable *ID*

Specifying a Ring

- Start with the smallest ID and list IDs in clockwise order.



- Example: 3, 37, 19, 4, 25

Uniform (Non-anonymous) Algorithms

- **Uniform** algorithm: there is one state machine for every id, no matter what size ring
- **Non-uniform** algorithm: there is one state machine for every id and every different ring size
- These definitions are tailored for leader election in a ring.

Overview of LE in Rings with IDs

- There exist algorithms when nodes have unique IDs.
- We will evaluate them according to their *message complexity*.
- asynchronous ring:
 - $\Theta(n \log n)$ messages
- synchronous ring:
 - $\Theta(n)$ messages under certain conditions
 - otherwise $\Theta(n \log n)$ messages
- All bounds are asymptotically tight.

The LCR algorithm

- Lelann-Chang-Robert (LCR) Algorithm, 1979
- The network graph is a directed ring (uni-directed or bi-directed) consisting of n nodes (n may be unknown to the processes ... Does this condition really require for rings?)
- Processes run the same deterministic algorithm
- The only piece of information supplied to the processes is a unique identifier (ID).
- IDs may be used
- In comparisons only (comparison-based algorithms)
- In comparisons and other calculations (non-comparison-based)

LCR algorithm - Description

- Each process sends its ID around the ring.
- When a process receives a ID, it compares this one to its own
- If the incoming ID is greater, then it passes this ID to the next process.
- If the incoming ID is smaller, then it discards it.
- If it is equal, then the process declares itself the leader.

LCR Algorithm for Leader Election

Alternative Algorithm:

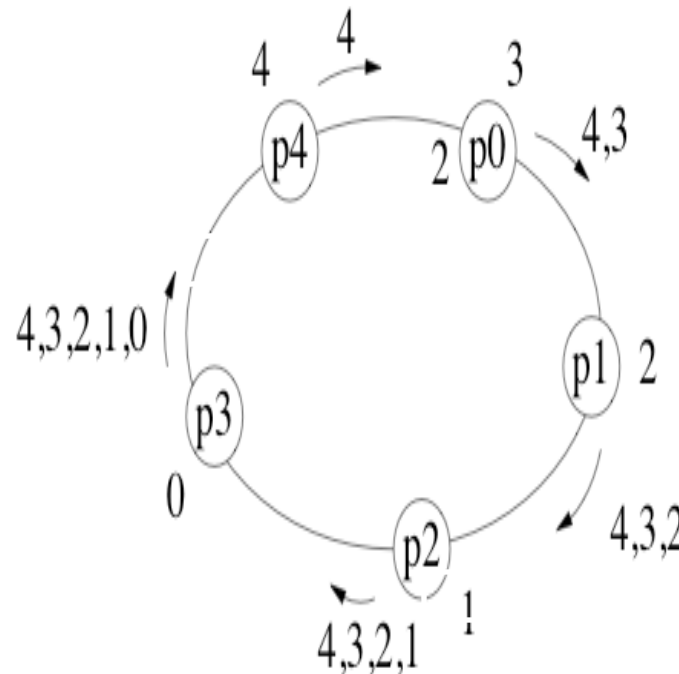
- send value of own id to the left
- when receive an ID j (from the right):
- if $j > \text{id}$ then
 - forward j to the left (this processor has lost)
- if $j < \text{id}$ then
 - do nothing
- if $j = \text{id}$ then
 - elect self as leader
(this processor has won)

Analysis of $O(n^2)$ Algorithm

- **Correctness:** Elects processor with largest id.
 - message containing largest id passes through every processor
- **Time:** $O(n)$
- **Message complexity:** Depends how the ids are arranged.
 - largest id travels all around the ring (n messages)
 - 2nd largest id travels until reaching largest
 - 3rd largest id travels until reaching largest or second largest
 - And so on

Analysis of $O(n^2)$ Algorithm

- Worst way to arrange the ids is in decreasing order:
 - 2nd largest causes $n - 1$ messages
 - 3rd largest causes $n - 2$ messages and so on
- Total number of messages is $n + (n-1) + (n-2) + \dots + 1 = \Theta(n^2)$.



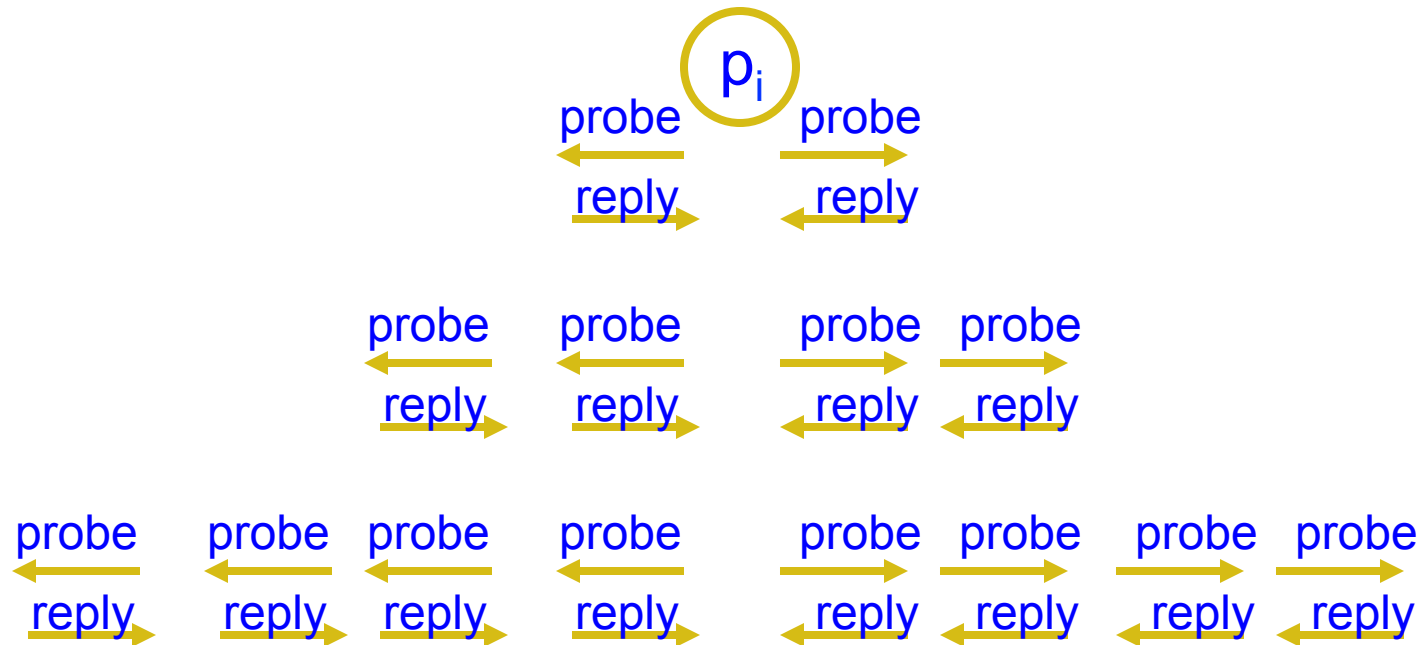
Can We Use Fewer Messages?

- The $O(n^2)$ algorithm is simple and works in both synchronous and asynchronous model.
- But can we solve the problem with fewer messages?
- Idea:
 - Try to have messages containing smaller ids travel smaller distance in the ring

$O(n \log n)$ Leader Election

- Each process tries to probe successively larger neighborhoods in both directions
 - size of neighborhood *doubles* in each phase
- If probe reaches a node with a larger id, the probe stops
- If probe reaches end of its neighborhood, then a reply is sent back to initiator
- If initiator gets back replies from both directions, then go to next phase
- If process receives a probe with its own id, it elects itself

$O(n \log n)$ Leader Election



Analysis of $O(n \log n)$ Algorithm

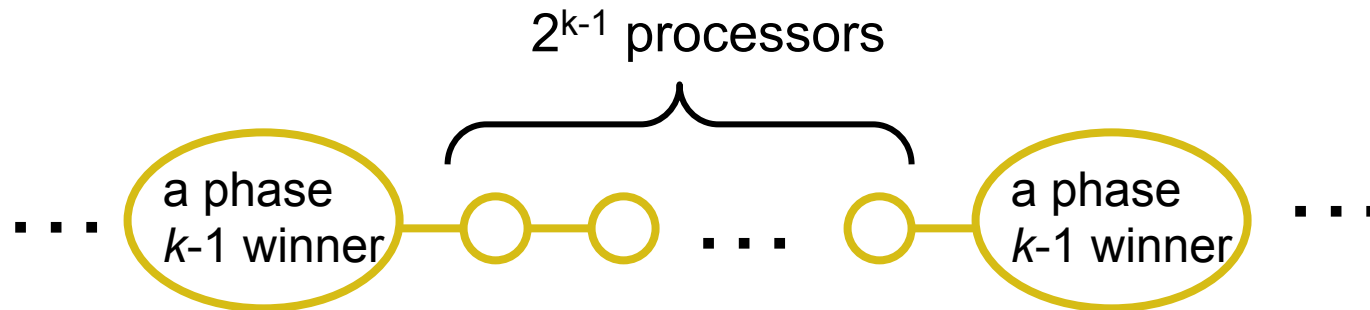
- **Correctness:** Similar to $O(n^2)$ algorithm.
- **Message Complexity:**
 - Each message belongs to a particular phase and is initiated by a particular proc.
 - Probe distance in phase k is 2^k
 - Number of messages initiated by a proc. in phase k is at most $4 \cdot 2^k$ (probes and replies in both directions)

Analysis of $O(n \log n)$ Algo

- How many processes initiate probes in phase k ?
 - For $k = 0$, every process does
 - For $k > 0$, every process that is a "winner" in phase $k - 1$ does
- "winner" means has largest id in its 2^{k-1} neighborhood

Analysis of $O(n \log n)$ Algo

- Maximum number of phase $k - 1$ winners occurs when they are packed as densely as possible:



- The total number of phase $k - 1$ winners is at most

$$n / (2^{k-1} + 1)$$

Analysis of $O(n \log n)$ Algo

- How many phases are there?
- At each phase the number of (phase) winners is cut approx. in half
 - from $n/(2^{k-1} + 1)$ to $n/(2^k + 1)$
- So after approx. $\log_2 n$ phases, only one winner is left.
 - more precisely, max phase is $\lceil \log(n-1) \rceil + 1$

Analysis of $O(n \log n)$ Algo

- Total number of messages is sum, over all phases, of number of winners at that phase times number of messages originated by that winner:

Diagram illustrating the analysis of the total number of messages:

$$\begin{aligned} & \text{phase 0 msgs} \rightarrow \leq 4n + n + \sum_{k=1}^{\lceil \log(n-1) \rceil + 1} 4 \cdot 2^k \cdot n / (2^{k-1} + 1) \\ & \text{termination msgs} \rightarrow \\ & < 8n(\log n + 2) + 5n \\ & = O(n \log n) \end{aligned}$$

msgs for phases 1 to $\lceil \log(n-1) \rceil + 1$

Can We Do Better?

- The $O(n \log n)$ algorithm is more complicated than the $O(n^2)$ algorithm but uses fewer messages in the worst case.
- Works in both synchronous and asynchronous case.
- Can we reduce the number of messages even more?
- Not in the asynchronous model ... !!

Summary

→ Leader Election

- Formulation of the problem

- LCR algorithm

- $O(n \log n)$ algorithm using probes

- Complexity Analysis

→ Many more to come up ... stay tuned in !!

How to reach me?

→ Please leave me an email:

rajendra [DOT] prasath [AT] iiits [DOT] in

→ Visit my homepage @

→ <http://www.iiits.ac.in/FacPages/index-rajendra.html>

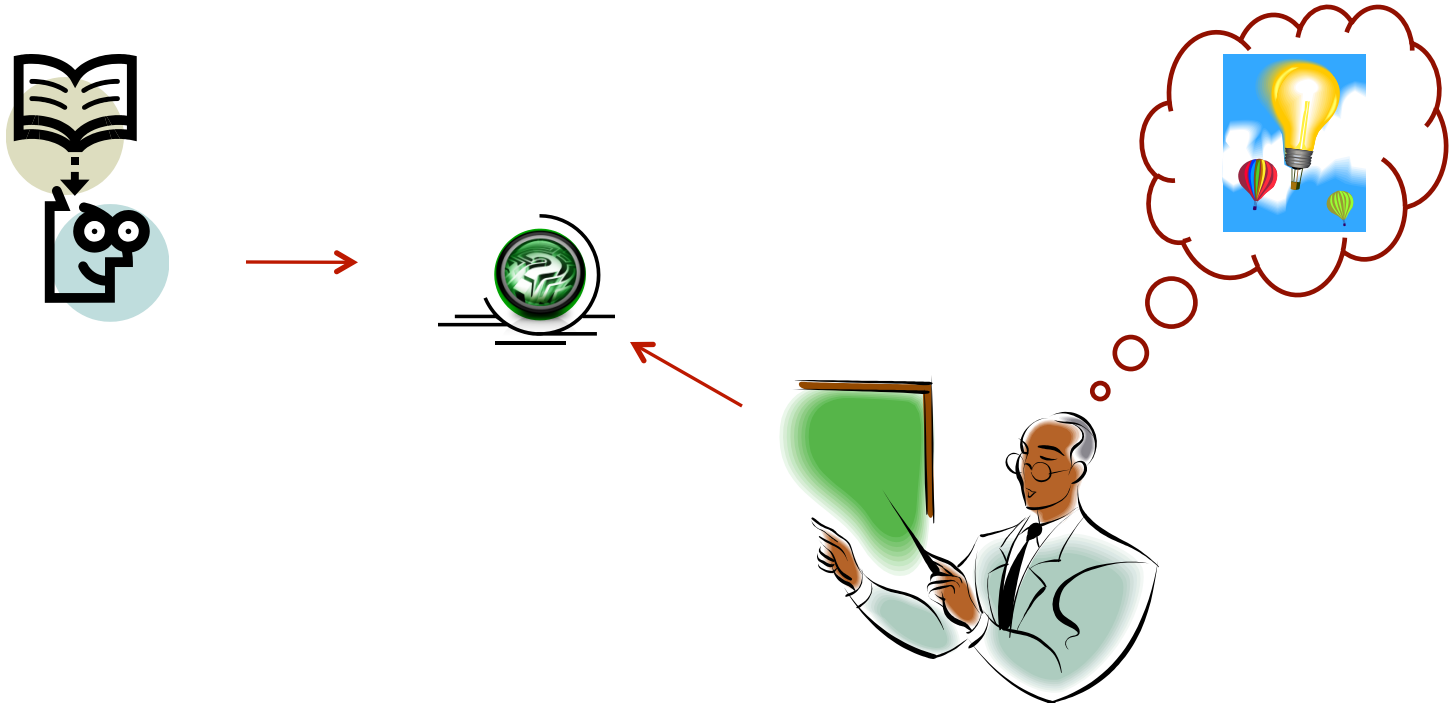
OR

→ <http://rajendra.2power3.com>

Help among Yourselves?

- **Perspective Students** (having CGPA above 8.5 and above)
- **Promising Students** (having CGPA above 6.5 and less than 8.5)
- **Needy Students** (having CGPA less than 6.5)
 - Can the above group help these students? (Your work will also be rewarded)
- You may grow a culture of **collaborative learning** by helping the needy students

Thanks ...



... Questions ???