Calvin is a transaction scheduling and replication layer provided above the storage system, which guarantees reduction in the contention costs in the distributed transactions in a distributed system. It does so by following a deterministic ordering. Most of the database systems trade off transaction processing for the linear scalability of the systems. Calvin achieves the goal of distributed transaction processing, along with scalability is by using a deterministic locking mechanism / protocol, which allows the elimination of distributed commit protocols.
By allowing distributed transactions, there is a possibility of deadlock in the system, which may lead to transactions being aborted and restarted, thereby reducing the throughput and the latency. Thus, to achieve the cheap distributed transactions and replications, Calvin performs in the following manner: multiple machines have an agreement on transaction handling before any of them acquires any locks or begins the transactions. These agreements are executed irrespective of any node failure or replication issues.
There are 3 layers in the Calvin, which scale horizontally, across cluster of nodes
Sequencing layer: Maintains a sequence or order of the transactions. This order is considered by each of the replicas, and also logs the input sequence.
Scheduling layer: Performs the transaction execution in the order specified by the sequencing layer, using a deterministic locking protocol.
Storage layer: It is used for the physical data layout of the application. Any storage engine can be used which supports CRUD operations interface, and it can be plugged in Calvin.

Strong Points:
1. Calvin removes long network round trip time consumed during distributed transactions in distributed systems, as each of the nodes make an agreement in advance, rather than making a transaction as in a traditional system.
2. Active replication allows the nodes to go to other replicas in case there are any crashes.
3. Calvin does periodic checkpoint of the systems by using 3 different checkpoint techniques, to provide a point for the recovery. Thus, by doing so during hardware failure, it recovers the system from recent points.

Weak Points:
1. One of the checkpoint techniques: naive synchronous checkpointing, can lead to lagging of any replica trying to checkpoint, compared to other replicas, and then this replica can raise problems of valid checkpoints, in case of system failure.
2. Whenever a replica fails, Calvin has to make other replicas wait for the node to fully recover from its recent snapshot, so that other nodes can read data., thereby affecting the throughput.
3. In a few instances of high contention, there is a slowdown in Calvin systems, compared to other systems.

Question:
How do the nodes understand the transactions of other nodes, while sending them for agreement?