1.0 Definition of Terms

Return to DIDO Consensus

The following definitions and explanations are provided to help explain DIDO **Consensus**.

Bizantine General Problem

Return to Top

TBD

- Byzantine Fault
- Byzantine Fault Tolerance (BFT)
- Byzantine Generals Problem

Consensus

Return to Top **Consensus**, in DIDOs, is when the entire distributed system agrees upon the state of the data within the system. In other words, the data within the entire system can be relied upon and reflects the *"truth"*. However, although the data within the DIDO is immutable, it does not mean it is static. Every proposed change in state to any data held within the DIDO is allowed when there is **Consensus** that the new data state is valid and verified.

- **Note:** There is a difference between a **DIDO Consensus** and a **Community of Interest (CoI) Consensus**.

DIDO Consensus

Return to Top

The **DIDO Consensus** is concerned with the way Consensus occurs to support the propagation of transactions throughout the DIDO Network. DIDO Consensus is generally inherent to the DIDO Platform. Any preference for a particular **Consensus Mechanism** on a particular project needs to be addressed as part of the functional requirements for the project. For example, there is a requirement for Proof of Work (PoW) or Proof of State (PoS).

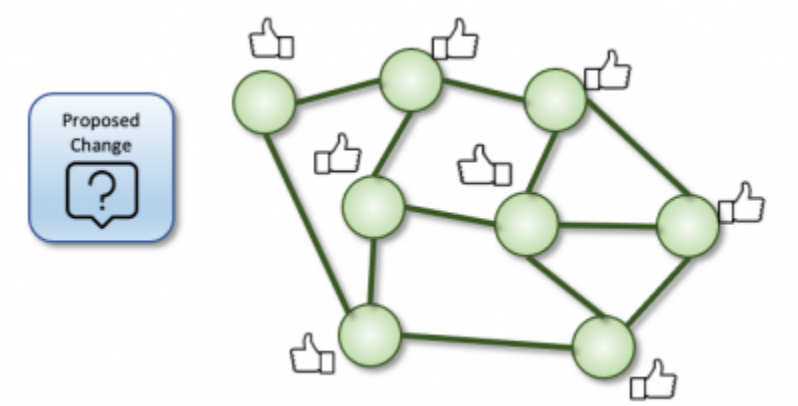


Figure 1: The DIDO Network Nodes have Consensus the data state represents *"truth"*.

CoI Consensus

Return to Top

CoI Consensus is concerned with how decisions are made in the CoI. The details of how Consensus is reached within a COI are generally captured in the Community's (i.e., Ecosphere's) Policies and Procedures (P&P).

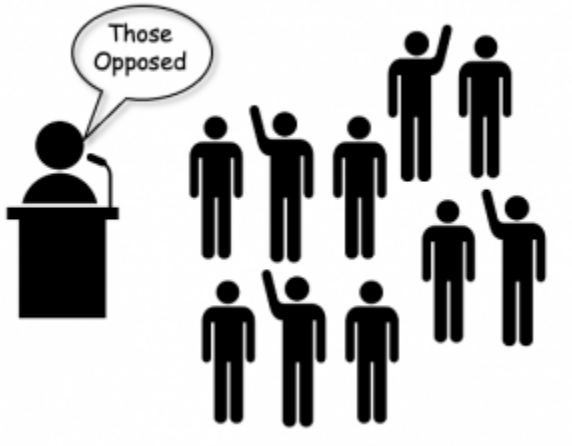


Figure 2: The COI Consensus.

- K.3 Consensus Mechanisms

Consensus Protocol

Return to Top

Consensus Protocol, in DIDOs, is developed by a specific DIDO platform to implement **Consensus Mechanism** over their DIDO network to achieve **Consensus**. This means that two DIDO Platforms can use the same Consensus Mechanism but have different implementations. The differences in the implementation makes a difference in the efficiency of the DIDO Platform and can be used as a selection

criterion between different Platforms. For example, there are two DIDO platforms (A and B) that both use Proof of Work(PoW). A is more efficient the B, therefore, the efficiency requirements will favor A.

- **Note:** There can be more evaluation criteria than efficiency.

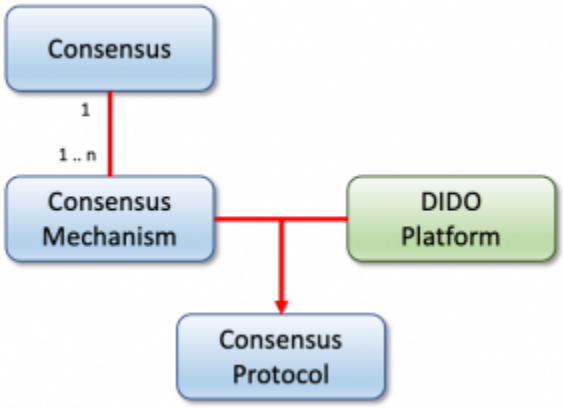


Figure 3: The relationship between Consensus, Consenus Mecahnism, and Consensus Protocol.

Transaction

Return to Top

All data within the DIDO are immutable with newer values (i.e., data states) being posted using a transaction that provides instructions on how to migrate the original data state to a new data state. Over time, the chain of history of data state changes (i.e., **Journal**) for one piece of data can become quite long. Given a known data state at a particular time, the current state of the data can be reconstructed using the journaled transactions.

In addition, the new data state also includes references back to the original data state. This allows the navigation of the journal back to the original data state (i.e., **Genesis Data**).

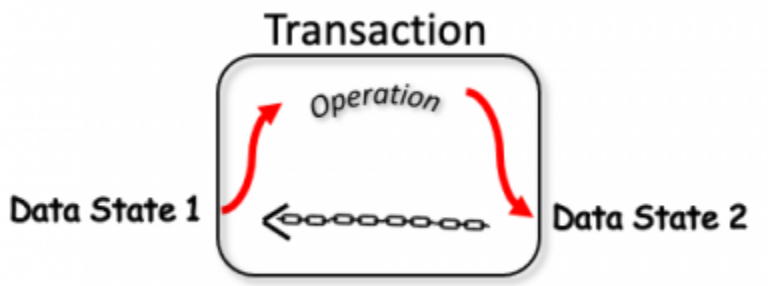


Figure 4: Transaction move the Data from one state to the next and remember the precious data state.

The following are some examples of Data State change commands. These are not Transactions because they do not include a reference to the original data to be changed (i.e., altered).

```
CHANGE EmotionState FROM HAPPY TO GLAD
CHANGE AccountBalance BY +5.00
```

- **Note:** In the example above, to be a transaction, the GLAD data state also has a reference back to

HAPPY data state.

Two Generals Problem

Return to Top

The **Two Generals' Problem** is an exercise illustrating the pitfalls and design challenges of coordinating action by communicating over an unreliable link. All too often, it is assumed that network links are available and are always available with little regard given to having unreliable network connectivity (also see Disconnected, Intermittent and Limited (DIL)).

- **Note:** This is related to, but distinct from the Bizantine General Problem. The Two Generals problem is about the weakness in the connectivity between the Generals (i.e., Node Network). The Byzantine General Problem is about the weakness of a General (i.e., in the Network Node.)

Problem Activities

Return to Top

In the exercise, two generals are only able to communicate with one another by sending a messenger through enemy territory. The purpose of the exercise is to coordinate the time of an attack on the common enemy fortress or castle.

1. Alpha General wants to send a message to Beta General that the attack is to occur at dawn
2. He formulates his message
3. Places the message in a sealed package
4. Hands it to the messenger that must travel through hostile enemy territory to deliver the message
5. The message is removed from its package and given to General Beta
6. General Beta agrees and sends an acknowledgment back to General Alpha
7. He formulates his message
8. Places the message in a sealed package
9. Hands it to the messenger that must travel back through hostile enemy territory to deliver the message
10. The message is removed from its package and given to General Alpha
11. General Alpha wants to ensure that the attack will occur at down, so he sends an acknowledgment he received General Beta's message by sending another acknowledgment back to General Beta
12. General Beta also wants to acknowledge General Alpha's acknowledgment, so he sends a message back.
13. And on and on and on until dawn and the attack

Since the two Generals can not speak directly to each other, there is a risk of failure at every step in the “message swap”. General Alpha has no idea that General Beta received the message until an acknowledgment is received. General Beta has no idea that General Alpha has received his acknowledgment. Since the defeat of the enemy requires both Generals and their troops, neither General

wants to commit to the attack unless they are sure of the joint attack.

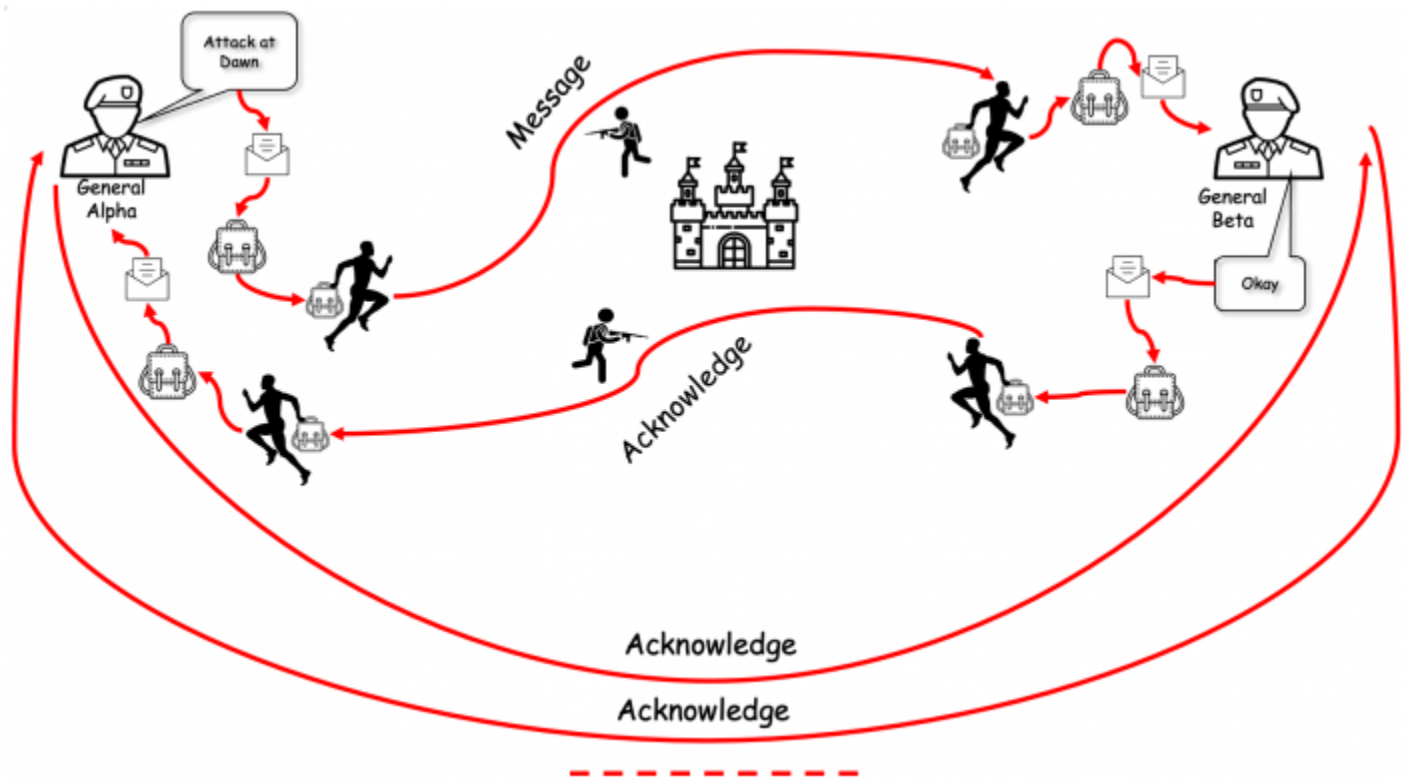


Figure 5: The Classic Two Generals Problem.

Conclusion

Return to Top

There is no definitive solution to the Two Generals' Problem. As a side effect, if the message contains a command that needs to be executed immediately, then submitting the message multiple times can cause duplicate commands. One way around this dilemma is using Idempotence which adds a unique number with the sending of each message. Every command message is checked when received for a previous command with the same unique number. If it was already received and process, then the duplicate command is ignored.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.k_consensus:01_definition:start&rev=1633293073

Last update: **2021/10/03 16:31**