A Compromise Between Synchronous and Asynchronous Systems

Tanisha Saxena

Mentor: Jun Wan

Distributed Systems

- Coordinating actions of separate networks by sending messages

- A universal problem is creating a network simulation that accurately represents what a network would act like with real people using it

- Examples: Telecommunication, World Wide Web, database management

Past Work On Async and Sync

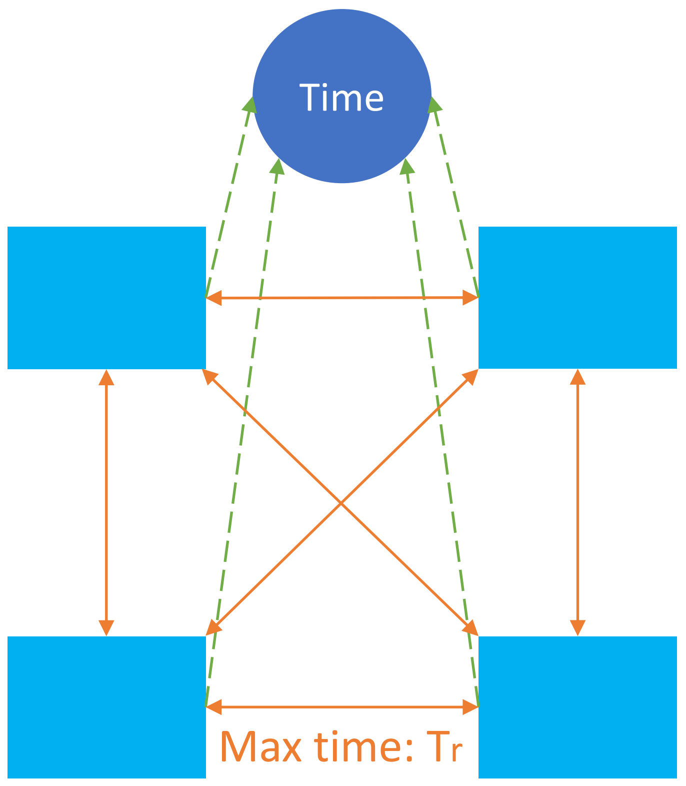
- Synchronous has many meanings, we only approach one

- Asynchronous can have many different restrictions, for this presentation we keep the simplest definition

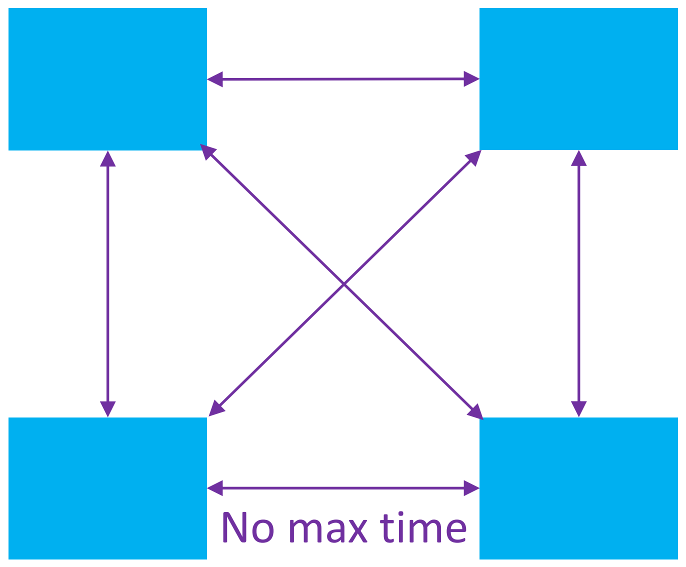
- There exist other models such as partially synchronous which combine features of both sync and async into one model but we are not focusing on these today

Background

Synchronous

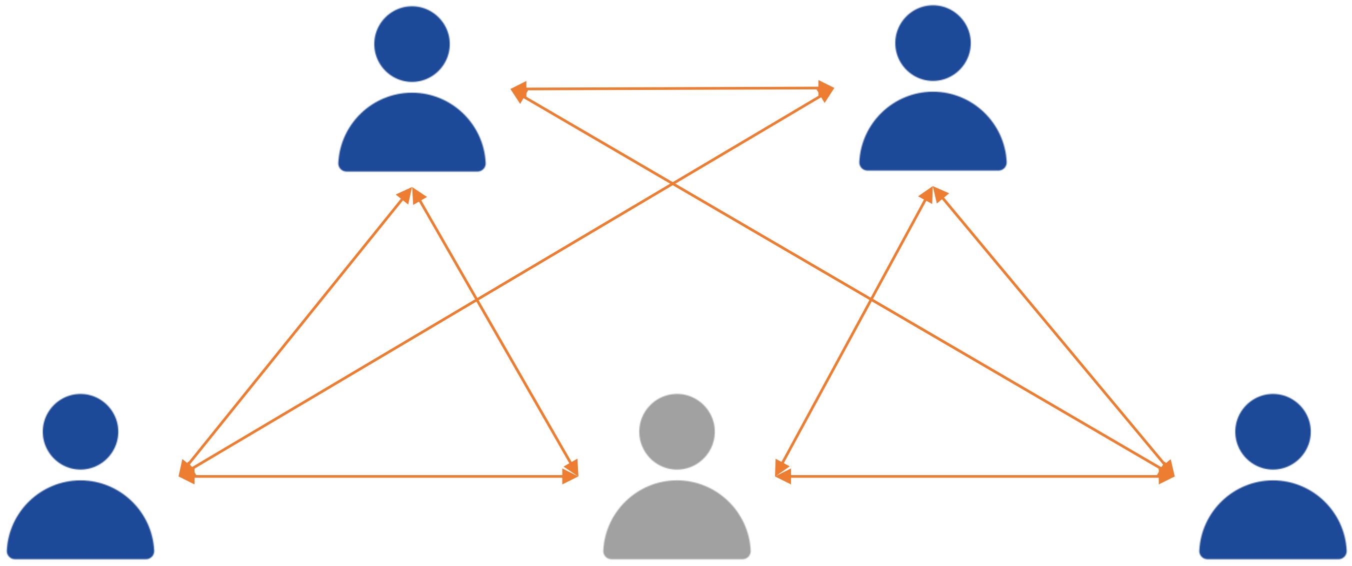


Asynchronous



Motivation

Create a model that accurately represents the synchronicity of real-world users in a system



Predefined Results

- The round model is a definition of a synchronous system where every action taken in round r should be completed before round r+1

- Two systems are equivalent if the respective pairs of matching users receive the same information in the same order such that the output of one system is a subset of the output of the other and vice versa

- Adversaries must have certain restrictions to prevent the system from never ending

Synchronous Model = Round Model

Reduction 2.3. *Given the definition of the synchronous and round model.*

1. $P(m, i, T_a)$: *reduces to* $P'(m, i, \left\lfloor \frac{T_a}{T_r} \right\rfloor)$. *In other word, on receiving a message* m *in time* T_a, P *simulates what* P' *would do if it receives* m *in round* $\left\lfloor \frac{T_a}{T_r} \right\rfloor$. *This ensures that any message sent in the sync model can be converted to an accurate counterpart in the round model.*

2. *Users only send message during the period* $T = 2k \cdot T_r, T = (2k+1) \cdot T_r$.

3. *Reaching end of time* T *event: if* $T = 2k \cdot T_r$ *for some constant* k, *then simulate what* P *would do at the end of round* k. *Otherwise, don't do anything.*

1. Using an integer $k \in \mathbb{Z}$, we can clarify time transformations between the time model and round model: in round k, any event that is taken within the round must end before the next round starts, equal to the time interval $(2kT_r, (2k+1)T_r)$ in the time model.

2. For event $R(A_i)$ in the time model where user A is sending a message at time t, this can be converted where $R(A_i) = (send, A, \lfloor \frac{t}{2T_r} \rfloor, m)$ where $2kT_r \leq t \leq (2k+1)T_r$, "send" is the action, and m is the message.

3. For event $R(B_j)$ in the time model where user B is receiving a message from user A at time t', $R(B_j) = (receive, B, A, \lfloor \frac{t'}{2T_r} \rfloor, m)$ where $t' < t + T_r \leq (2k+2)T_r$.

4. The described restrictions for t' also mean that $\lfloor \frac{t'}{2T_r} \rfloor < k+1$, therefore, $R(B_j) = (deliver, B, A, k, m)$.

5. This proves that the events causing a message sent at time t and received at t' all take place within the same round k, meaning any message events in the time model can be converted to the round model while fitting the respective restrictions.

Synchronous Model = Round Model Intuition

- We create an equation to do a time conversion between the global time T of the sync model and round R of the round model
- Show how times of events (send, receiving, round ending/starting) can all be converted between the two
- Sequence and content of events is the same so models are equal

Review the Differences

- Synchronous
  - Global clock
  - Restricted delivery time

- Asynchronous
  - No Global Clock
  - Unrestricted delivery time

Asynchronous with Clock

- Compare an asynchronous system with synchronous clocks to a regular asynchronous system

- Help determine what factors are important to creating a slightly stronger system

Define: An asynchronous system with synchronous clocks is one where each user can access a global clock with no time cost, but the delivery time is unrestricted with only the guarantee that a message will eventually be received.

Proving Global Time is Irrelevant

Reduction 3.3. *Given: We create a global synchronous time for the asynchronous system that can be accessed with no time delay by every user. This is compared to the asynchronous + sync clock system described earlier.*

1. *Sending a message* $P(A, m, T_s)$ *can be converted to sending a message* $P'(A, m, T_a)$ *where* $T_a = T_s$.
2. *When user* B *receives the message in* P', *its local clock is changed such that* $T_b = T_a + r$ *where* $r > 0$ *and* $r \in \mathbb{Z}$.
3. *Hence the message that is stored as* $P(A, m, T_s, T_r)$ *can be converted to* $P'(A, m, T_a, T_b)$ *after* T_b *has been adjusted such that* $T_b > T_a$ *as described earlier,*
4. *The adjustment of clocks in* P' *ensures that all saved events of the protocol match directly with the events in* P, *as a message sent at time* T_a *is always received after time* T_a *and is therefore stored in the correct order.*
5. *Since all events in both protocol occur in the same order, they are proven to be equivalent.*

Intuition Summary

- The internal independent clocks of the async without clocks system add random values after receiving messages to simulate the passage of time relative to the user it received from

- This ensures all messages are received in the same sequence and that the right number of messages is received before the system ends

Introducing Adversaries

- Use adversaries to change the actions of a select few users

Define: A normal adversary allows up to f users to send arbitrary messages at any time

Define: A special adversary does what a normal adversary does and also, for any number of honest users, it can block messages from up f users

After Adding Adversaries

- After this, we came to the conclusion that
    - Synchronous + Special Adversary = Asynchronous + Normal Adversary
- Adding adversaries was useful in showing which parts of a system can add and remove synchronicity
- Synchronous + Alternate adversary could be close to an accurate model

Conclusion

- Changing the delivery time and messages holds the biggest bearing over the strength of the system
- Create more adversaries to create more interesting effects on systems
- Find an adversary that defines a system closest to the real world

Thank You

- My mentor, Jun Wan
- Srini Devadas, Slava Gerovitch, MIT PRIMES Program, for this opportunity
- My family

References

[CD+15] Ronald Cramer, Ivan Bjerre Damgard, et al. Secure multiparty computation. Cambridge University Press, 2015.

[Cri91] Flaviu Cristian. “Reaching agreement on processor-group membership in synchronous distributed systems”. In: Distributed Computing 4.4 (1991), pp. 175–187.

[Del+07] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Bastian Pochon. “The perfectly synchronized round-based model of distributed computing”. In: Information and Computation 205.5 (2007), pp. 783–815.

[MMR14] Achour Mostefaoui, Hamouma Moumen, and Michel Raynal. “Signature-free asynchronous Byzantine consensus with t¡ n/3 and O (n2) messages”. In: Proceedings of the 2014 ACM symposium on Principles of distributed computing. 2014, pp. 2–9.