A Systematic Study on the Difference and Conversion Between Synchronous and Asynchronous Protocols

Tanisha Saxena

Mentor: Jun Wan

MIT PRIMES

2022 May 22nd Conference

Background

- **Distributed systems**: A network of *n* users that communicate by sending messages to each other.

- **Synchronous systems**: Delivery-time-bounded systems where each user can access the global clock with zero delay.

- **Asynchronous systems**: Systems in which users only have access to local clocks and messages are not bounded.

Last Year’s Work

- Find a distributed systems model similar to the real world
- Aimed to combine synchronous and asynchronous models
- Created an adversary that got close to the goal

Motivation and Goal

Model	Practical	Efficient	Simple
Synchronous	No	Yes	Yes
Asynchronous	Yes	No	No

Developers need a method to create less complex protocols that work on real systems

Goal

Identify conditions under which a synchronous protocol can be converted to an asynchronous protocol

Equivalence of Protocols

A distributed protocol can be understood as a function *P(i)* where *i* is an input vector containing the input messages for each user. The protocol creates an output vector *o* that specifies the output generated by each user. Two protocols are equivalent if their output vectors are identical.

General Algorithm

$P(i) \rightarrow o$

$P'(i) \rightarrow o'$

$o = o'$

Distributed Algorithm

$P(i, \alpha) \rightarrow o$

$P'(i, \alpha') \rightarrow o'$

$\forall i, \alpha \ \ \exists \alpha' \ni o = o'$

Clock in Asynchronous Model

We begin by analyzing the impact of the “global clock” part of the synchronous model. To do this, we add a global clock to the asynchronous model, and observe how that effects the users.

Result

An asynchronous model with a global clock is equivalent to the plain asynchronous model

Therefore, the global clock alone does not add synchronicity to the asynchronous model.

Redefining Protocols

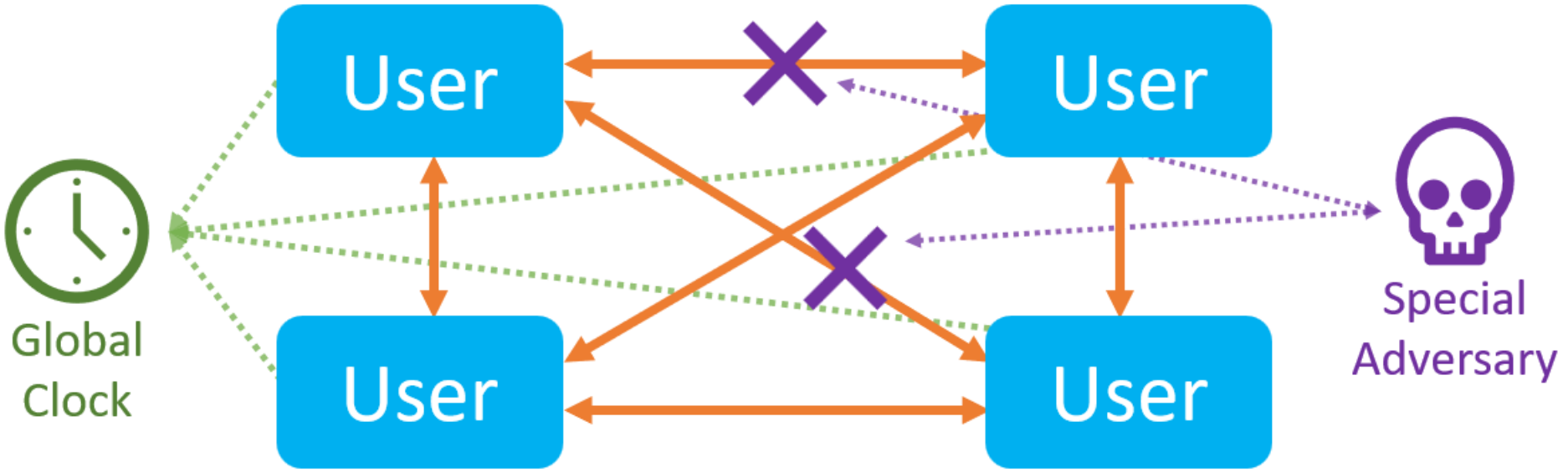
To improve the simplicity of defining protocols, we prove the following:

- Time-based synchronous model = Round-based synchronous model (known)
- Any protocol can be described as a sequence of events

The event-order-based description of the protocols makes it much easier for us to prove equivalence, and makes it easier to define adversaries

Adversary in Synchronous Model

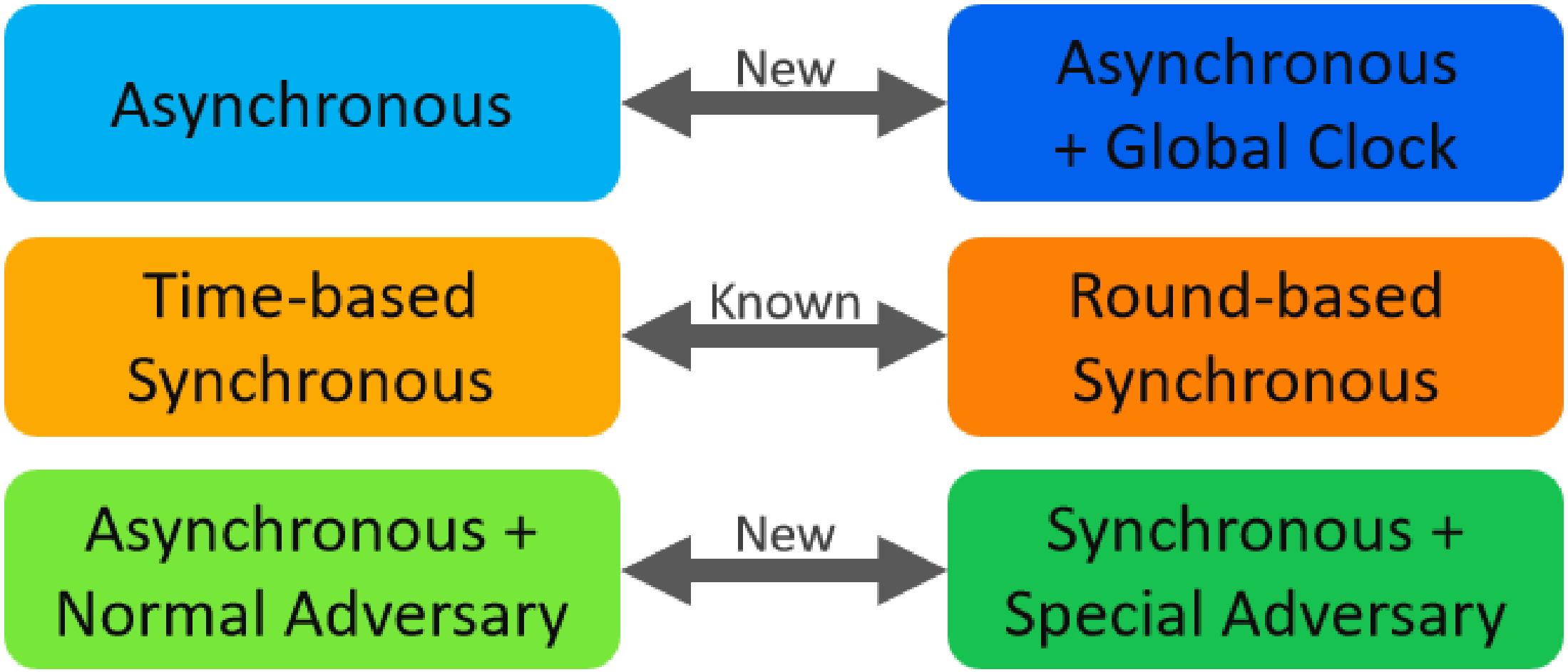
- A **normal adversary** can corrupt up to *f* users, and allows them to send any arbitrary message at any time.
- A **special adversary** can, in addition to the effects of the normal adversary, block up to f messages from other users



Result

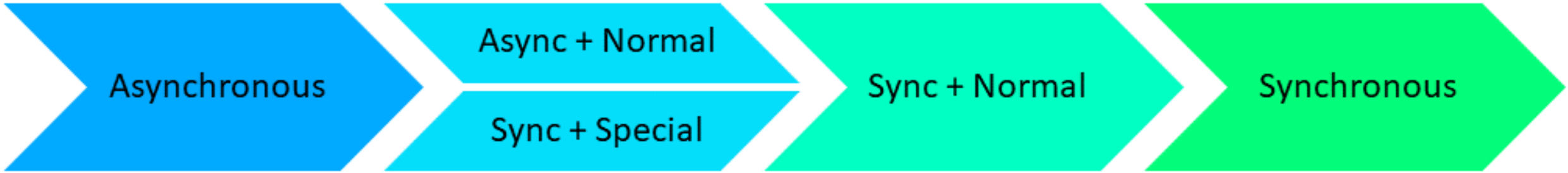
A synchronous system with a special adversary is equivalent to an asynchronous system with a normal adversary

Equivalence Results



Hierarchy Results

Our research along with our special adversary provide us with a method to order distributed system models based on how restrictive they are:



This allows for more precise navigation of synchronicity in distributed systems overall.

Conclusion

- We determined the extent of the effect of the main restrictions on distributed systems
  - Global clock
  - Time-bounded delivery
- Our special adversary combined with the synchronous system acts as a structured model to create simpler protocols
  - Allows for transformation between synchronous and asynchronous protocols
- Our hierarchy allows developers to precisely alter the synchronicity of their models to accurately simulate any distributed system

Future Work

- Can we retain the model hierarchy after removing the normal adversary?
- Does the normal adversary effect the efficiency of users in the asynchronous model?
- Can we generalize adversary effects to create models of any arbitrary strength?
- Other than the protocols we have already tested, which other distributed systems protocols can be converted to synchronous or asynchronous given our models?

Acknowledgements

- Jun Wan, my mentor, for his invaluable insight and guidance
- Srini Devadas for supporting me throughout the research journey
- Elaine Shi for providing critical feedback on application of our work
- Yu Xia for his generous comments on how to develop our work
- Slava Gerovitch and the entire MIT PRIMES program for giving me this opportunity

Bibliography

- [BFA21] Raul Barbosa, Alcides Fonseca, and Filipe Araujo. “Reductions and abstractions for formal verification of distributed round-based algorithms”. In: Software Quality Journal (2021), pp. 1–27.
- [Can+96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. “Adaptively secure multi-party computation”. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996, pp. 639–648.
- [CD+15] Ronald Cramer, Ivan Bjerre Damg̊ard, et al. Secure multiparty computation. Cambridge University Press, 2015.
- [CL85] K Mani Chandy and Leslie Lamport. “Distributed snapshots: Determining global states of distributed systems”. In: ACM Transactions on Computer Systems (TOCS) 3.1 (1985), pp. 63–75.
- [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 tɟ n/3 and O (n2) messages”. In: Proceedings of the 2014 ACM symposium on Principles of distributed computing. 2014, pp. 2–9.