一种兼具分区容错的弱同步模型

同步模型

定义 诚实的节点发送一个消息，诚实的接收方一定能在一个具有上限的时间延迟Δ内接受到此消息。

优点

- 在同步模型的假设下，系统的鲁棒性能够很好地得到保证。
- 即使系统内任意多节点是恶意的，那么系统也总能达成分布式共识。

局限

- 但message延迟任意长的时间，在系统内存在三分之一的节点是腐败的情况下，分布式共识是不可能达到的。
- 另外，经典同步的模型假设条件太严苛：

  如果一个节点出现非常短暂的outage（比如network jitter），那么在规定的延迟时间内这个节点将无法接受到诚实节点发送的message，那么此节点会被认为是腐败的，即恶意节点。
- 因此，在经典同步模型下，共识协议无法再为此类节点保证一致性与活性，即使这个节点很快复活并希望重新加入协议；多方安全计算协议将无法再为P方保证其隐私性，因为协议会令其余在线各方重构P的秘密输入以保证公平性以及输出

如何改进同步模型？

常用方法

- 通常解决上述问题的方法是利用部分同步或者异步模型。
- 部分同步模型可以任意多容错，异步模型不能容错，两者最多容忍三分之一的腐败节点（恶意节点）。

能否设计一种可以容忍超过三分之一腐败节点，且同时具备容错能力的分布式计算协议？

$\chi-$ Weak Synchronous

同步和分区容错并非是一个非0即1的问题，这篇文章通过一种量化的同步操作确保了一个“最佳概率分区容错（best-possible partition tolerance）”概念。也提出了一种新的模型，叫做$\chi-$**弱同步模型。**这个模型可以看作是经典同步模型（或强同步模型）的一般化，考虑到了网络的搅动。

$\chi-$ **弱同步模型假设**：在协议的每一轮中，会有超过χ倍的节点是诚实并且在线的。但是相邻的每一轮中，这些诚实且在线的节点未必都是相同的。

O_r：记做在第r轮中，诚实且在线的节点集合。每轮诚实且在线的节点个数至少为$\lfloor \chi n \rfloor + 1$。

网络假设：在O_r 中的节点在第r轮发送了一个消息，那么O_t 中的节点，$t \geq r + \Delta$，一定能在第t轮收到这个消息。

挑战：1.在相邻的两轮中，在线节点的变化速度非常快；
2.节点意识不到自己是否在线或者掉线；
3.恶意节点会故意泄露一些消息的子集合，用以蒙蔽掉线的诚实节点，是他们无法检测到自己已掉线。

Consensus in a Weakly Synchronous Network

分布式计算的主要应用场景包括共识算法和多方安全计算，这篇文章主要考虑的是在弱同步模型下实现**拜占庭共识协议**。

拜占庭协议性质：

1. 一致性（consisitency）：所有诚实的节点必须输出一致比特。
2. 有效性（validity）：如果被指定的发送方是诚实并且在线的在协议初始的第0轮，那么其余每个诚实节点的输出必须认可发送方的输入比特。
3. T-liveness：每个O_r中的节点（r≥t）在第r轮结束时肯定会产生一个输出。

考虑到要设计一个能容忍大于三分之一腐败的同步模型，为了保证χ的下限，弱同步模型一律假设PKI的存在。事实证明，χ不能小于0.5，即在诚实且在线的节点个数是少数时，无法实现弱同步网络。

Best-possible partition tolerance：在0.5-弱同步模型下，一个具备一致性、有效性和T-liveness的共识协议也是Best-possible partition tolerance的。

注：任何一个Best-possible partition tolerance的拜占庭协议（在0.5-弱同步模型下时安全的）那么在强同步模型下也是安全的，反之不成立。

Defining a Weakly Synchronous Execution Model

协议的执行被看作是交互式图灵机（ITM）的集合，定义一个非均匀的概率多项式时间环境Z（1^k）。协议中的节点由ITM充当，而节点的个数1到n(κ)，n由Z（1^k）指定。

敌手A（1^k）：非均匀的概率多项式算法。A可以和Z（1^k）无限次的任意交流，所有腐败节点均有A掌控。协议开始前，敌手首先指定好哪些节点是腐败的，所以这里的腐败节点是静态的。

Assumption 1 (**Message delivery assumption**). *We assume that if someone in* $\mathcal{O}_r$ *multicasts a message* m *in round* r*, then everyone in* $\mathcal{O}_t$ *where* $t \geq r + \Delta$ *will have received* m *at the beginning of round* t.

Definition 2 (χ**-weak-synchrony**). *We say that* $(\mathcal{A}, \mathcal{Z})$ *respects* χ*-weak-synchrony (or that* $\mathcal{A}$ *respects* χ*-weak-synchrony), iff in every round* r, $|\mathcal{O}_r| \geq \lfloor \chi \cdot n \rfloor + 1$.

敌手服从假设1，那么意味着敌手可以任意擦除或更改信息。敌手服从x-Weakly Synchronous，意味着敌手最多控制$\lfloor \chi n \rfloor - 1$ 个节点。

Defining a Weakly Synchronous Execution Model

敌手$A(1^k)$：非均匀的概率多项式算法。A可以和$Z(1^k)$无限次的任意交流，所有腐败节点均有A掌控。假设PKI和计算能力有上限的敌手是存在的。

1. 每个诚实节点都从Z接收输入，并从网络接收传入消息；请注意，此刻，A决定一个诚实节点接收哪一组传入消息将影响该诚实节点是否可以包含在 O_r中；
2. 然后，每个诚实节点执行多项式时间的计算，并确定要发送到其他节点的消息-这些消息会立即显示给A。此外，在计算之后，每个诚实节点都可以选择将输出发送给Z。
3. 此时，A决定哪些节点属于O_r ，其中r表示当前回合。请注意，A可以在查看诚实节点打算在此回合中发送哪些消息之后，决定本回合的诚实和在线集Or。
4. 现在，A决定每个损坏的节点将发送到每个诚实节点的消息。还要注意的是，A是占有先机的，因为在决定损坏节点的消息之前，它可以看到所有诚实消息。
5. 诚实节点通过网络将消息发送到其他节点（只要满足假设1，它们可以被A延迟或擦除）。

Defining a Weakly Synchronous Execution Model

协议的执行被看作是交互式图灵机（ITM）的集合，定义一个非均匀的概率多项式时间环境Z（1^k）。协议中的节点由ITM充当，而节点的个数1到n(κ)，n由Z（1^k）指定。

P ∈{consistency, validity, T-liveness}，若一个BA协议满足P，可写作任一服从χ- 弱同步模型的非均匀概率多项式时间(A,Z)，在共享同一PKI下，被允许可以生成多个可能并行的拜占庭协议实例。

- *Consistency.* If honest node i outputs b_i and honest node j outputs b_j, it must be that $b_i = b_j$.
- *Validity.* If the sender is in $\mathcal{O}_0$, any honest node's output must be equal to the sender's input.
- *T-liveness.* Any node in $\mathcal{O}_r$ for $r \geq T$ must have output a bit by the end of round r.

how to realize BA under 0.5-weak synchrony ?

技术路线：

Reliable Broadcast (RBC) ⇒ Verifiable Secret Sharing (VSS) ⇒ Leader Election (LE) ⇒ Byzantine Agreement (BA)

可信广播

Syntax. An RBC protocol consists of the following algorithms/protocols:

- **PKI setup:** at the very beginning every node i registers a public key pk_i with the PKI;
- RBC **protocol:** all instances of RBC share the same PKI. In each RBC instance, a designated sender (whose identifier is pre-determined and publicly-known) receives a value x from the environment $\mathcal{Z}$ whereas all other nodes receive nothing. Whenever a node terminates, it outputs a value y. Henceforth we shall assume that an admissible $\mathcal{Z}$ must instruct all nodes to start protocol execution in the same round[2];
- **Extractor** $\mathcal{E}$: a polynomial-time deterministic extractor $\mathcal{E}$ that is needed only in our security definitions and proofs, not in the real-world protocol.

how to realize BA under 0.5-weak synchrony ?

可信广播

Security. Let $T(n, \Delta, \kappa)$ be a polynomial function in the stated parameters. For $P \in \{T\text{-consistency, validity, } T\text{-liveness, close termination}\}$, we say that an RBC protocol Π satisfies property P under χ-weak-synchrony iff for any non-uniform p.p.t. $(\mathcal{A}, \mathcal{Z})$ that respects χ-weak-synchrony and can spawn multiple instances of RBC sharing the same PKI, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\kappa \in \mathbb{N}$, except for $\mathsf{negl}(\kappa)$ fraction of the executions in the experiment $\mathsf{EXEC}^{\Pi}(\mathcal{A}, \mathcal{Z}, \kappa)$, the following properties hold for every RBC instance:

how to realize BA under 0.5-weak synchrony ?

可信广播——性质

- *T-consistency.* Let $y := \mathcal{E}(\{\mathsf{pk}_i\}_{i\in[n]}, \mathsf{Tr})$ where Tr denotes the transcript of all honest nodes in the initial T rounds of the RBC instance. Then, if any honest node ever outputs y', it must be that $y' = y$.
- *Validity.* If the sender is honest and its input is x, then if any honest node outputs x', it must be that $x' = x$.
- *T-liveness (under an honest and initially online sender).* If the sender is not only honest and but also online in the starting round of this RBC instance (henceforth the starting round is renamed to be round 0 for convenience), then every node that is honest and online in round $r \geq T$ will have produced an output by the end of round r.
- *Close termination.* If an honest node outputs in some round r, then every node that is honest and online in round $r' \geq r + 2\Delta$ will have output by the end of round r'.

how to realize BA under 0.5-weak synchrony ?

可验证的秘密共享：

允许一个dealer在所有节点间共享秘密，并随后可以重构这个秘密。

PKI setup (shared across all VSS instances)**:** During the PKI setup phase, every node i performs the following:

- let $(\mathsf{epk}_i, \mathsf{esk}_i) \leftarrow \mathsf{PKE.K}(1^\kappa)$; $(\mathsf{vk}_i, \mathsf{ssk}_i) := \Sigma.\mathsf{K}(1^\kappa)$; $\mathsf{crs}_i \leftarrow \mathsf{NIZK.K}(1^\kappa)$; and let $(\mathsf{rpk}_i, \mathsf{rsk}_i) \leftarrow \mathsf{RBC.K}(1^\kappa)$;
- node i registers its public key $\mathsf{pk}_i := (\mathsf{epk}_i, \mathsf{crs}_i, \mathsf{vk}_i, \mathsf{rpk}_i)$ with the PKI; and it retains its secret key comprised of $\mathsf{sk}_i := (\mathsf{esk}_i, \mathsf{ssk}_i, \mathsf{rsk}_i)$.

Share (executed by the dealer)**:** Let s be the input received from the environment, the dealer does the following:

- it splits s into n shares using a $(\lfloor n/2 \rfloor + 1)$-out-of-$n$ Shamir Secret Sharing scheme, where the i-th share is henceforth denoted s_i;
- for $i \in [n]$, it computes $\mathsf{CT}_i := \mathsf{PKE.Enc}_{\mathsf{epk}_i}(sid, s_i)$ where sid is the identifier of the current instance;
- it calls $\mathsf{NIZK.P}(\{\mathsf{crs}_i\}_{i\in[n]}, x, w)$ to compute a proof π where x and w are defined as below: $x := (sid, \{\mathsf{pk}_i, \mathsf{CT}_i\}_{i\in[n]})$ is the statement declaring that there is a witness $w := (s, \{s_i\}_{i\in[n]})$ such that for each $i \in [n]$, CT_i is a valid encryption[5] of (sid, s_i) under epk_i (which is part of pk_i); and moreover, the set of shares $\{s_i\}_{i\in[n]}$ is a valid sharing of the secret s.
- finally, the dealer relies on RBC to reliably broadcast the tuple $(sid, \{\mathsf{CT}_i\}_{i\in[n]}, \pi)$—henceforth this RBC instance is denoted RBC_0.

how to realize BA under 0.5-weak synchrony ?

技术路线：

Reliable Broadcast (RBC) ⇒ Verifiable Secret Sharing (VSS) ⇒ Leader Election (LE) ⇒ Byzantine Agreement (BA)

可验证秘密共享

Share (executed by everyone)**:** Every node i does the following (where the starting round of Share is renamed round 0):

- **Any time:** whenever the RBC_0 instance outputs a tuple of the form $(sid, \{\mathsf{CT}_j\}_{j\in[n]}, \pi)$, call NIZK.V to verify the proof π w.r.t. the statement $(sid, \{\mathsf{pk}_i, \mathsf{CT}_i\}_{i\in[n]})$; and if the check succeeds, set flag := 1 (we assume that flag was initially 0).
- **Round** T_{rbc}**:** if flag $= 1$, reliably broadcast the message "ok"; else reliably broadcast the message "$\perp$";
- **Any time:** whenever more than $\lfloor n/2 \rfloor + 1$ RBC instances have output "ok" and RBC_0 has output a tuple; decrypt CT_i contained in the tuple output by RBC_0 using secret key esk_i; let $(\_, s_i)$ be the decrypted outcome; now record the share s_i and output "sharing-succeeded";

Reconstruct (executed by everyone)**:** when the Reconstruct sub-protocol has been invoked, every node i waits till the instance's Share sub-protocol has output "sharing-succeeded" and then performs the following where the set $\mathbb{S}$ is initially empty:

- let s_i be the share recorded at the end of the Share sub-protocol;

how to realize BA under 0.5-weak synchrony ?

技术路线：

Reliable Broadcast (RBC) ⇒ Verifiable Secret Sharing (VSS) ⇒ Leader Election (LE) ⇒ Byzantine Agreement (BA)

可验证秘密共享

- call $\mathsf{NIZK.P}(\{\mathsf{crs}_i\}_{i\in[n]}, x, w)$ to compute a proof (henceforth denoted π_i) for the following statement $x := (sid, i, s_i, \mathsf{CT}_i)$ declaring that there is random string that causes $\mathsf{PKE.K}$ to output the tuple $(\mathsf{epk}_i, \mathsf{esk}_i)$ where $\mathsf{epk}_i \in \mathsf{pk}_i$; and moreover, (sid, s_i) is a correct decryption of CT_i using esk_i—the witness w includes the randomness used in $\mathsf{PKE.K}$, esk_i, and the randomness of $\mathsf{PKE.Dec}$.
- multicast the tuple (sid, i, s_i, π_i);
- upon receiving a tuple (sid, j, s_j, π_j) such that π_j verifies w.r.t. the statement $(sid, j, s_j, \mathsf{CT}_j)$ where CT_j was the output of RBC_0 during the Share sub-protocol, add s_j to the set $\mathbb{S}$.
- whenever the set $\mathbb{S}$'s size is at least $\lfloor n/2 \rfloor + 1$, call the reconstruction algorithm of Shamir Secret Sharing to reconstruct a secret s, and if reconstruction is successful, output the result.

how to realize BA under 0.5-weak synchrony ?

技术路线：

Reliable Broadcast (RBC) ⇒ Verifiable Secret Sharing (VSS) ⇒ Leader Election (LE) ⇒ Byzantine Agreement (BA)

领导人选举（LE）

- **PKI setup** (shared across all LE instances)**:** each node i calls $(\mathsf{rpk}_i, \mathsf{rsk}_i) \leftarrow \mathsf{RBC.K}(1^\kappa)$; $(\mathsf{vpk}_i, \mathsf{vsk}_i) \leftarrow \mathsf{VSS.K}(1^\kappa)$; and $(\mathsf{vk}_i, \mathsf{ssk}_i) \leftarrow \Sigma.\mathsf{K}(1^\kappa)$. Now its public key is $(\mathsf{rpk}_i, \mathsf{vpk}_i, \mathsf{vk}_i)$ and its secret key is $(\mathsf{rsk}_i, \mathsf{vsk}_i, \mathsf{ssk}_i)$.
  In the following, we describe the leader election (LE) protocol. We assume that all LE protocols share the same PKI. Moreover, whenever a node i uses ssk_i to sign messages, the message to be signed is always tagged with the session identifier sid of the current instance and signature verification also verifies the signature to the same sid.
- **Round 0:** Node i chooses n random coins $c_{i,1}, \ldots, c_{i,n} \in \mathbb{F}$. For instances $\mathsf{VSS}[i,1]$, ..., $\mathsf{VSS}[i,n]$ where node i is the dealer, node i provides the inputs $c_{i,1}, \ldots, c_{i,n}$ respectively to each instance. Then, node i invokes the Share sub-protocol of all n^2 instances of VSS.
- **Any round:** At any time during the protocol, if in node i's view, all n VSS instances where node j is the dealer has terminated outputting "`sharing succeeded`", we say that node i now considers j as a *qualified dealer*.

how to realize BA under 0.5-weak synchrony ?

领导人选举（LE）

- **Round T_{vss}:** If in round T_{vss}, at least $\lfloor n/2 \rfloor + 1$ qualified dealers have been identified so far: let D be the current set of all qualified dealers; reliably broadcast the message (`qualified-set`, D) using RBC. Henceforth, we use RBC$[j]$ to denote the RBC instance where j is the sender. If not enough qualified dealers have been identified, reliably broadcast the message $\perp$.
- **Any round:** In any round during the protocol, if RBC$[j]$ has output (`qualified-set`, D_j) such that D_j is a subset of $[n]$ containing at least $\lfloor n/2 \rfloor + 1$ nodes, and moreover every node in D_j has become qualified w.r.t. node i's view so far, then node i considers j as a *candidate*, and node i records the tuple (j, D_j).
- **Round $T_{\text{vss}} + T_{\text{rbc}}$:** In round $T_{\text{vss}} + T_{\text{rbc}}$, do the following:
  - invoke the Reconstruct sub-protocol of all VSS instances;
  - if at least $\lfloor n/2 \rfloor + 1$ nodes are now considered candidates: let S be the set of all candidates so far; now multicast (`candidate-set`, S) along with a signature on the message.
- **Any round:** At any time, if a node i has observed a (`candidate-set`, S_j) message with a valid signature from the purported sender j where $S_j \subseteq [n]$ is at least $\lfloor n/2 \rfloor + 1$ in size, and moreover, every node in S_j is now considered a candidate by node i too, we say that node i becomes *happy* with j.

how to realize BA under 0.5-weak synchrony ?

领导人选举（LE）

– **Round T_{vss}:** If in round T_{vss}, at least $\lfloor n/2 \rfloor + 1$ qualified dealers have been identified so far: let D be the current set of all qualified dealers; reliably broadcast the message (`qualified-set`, D) using RBC. Henceforth, we use RBC$[j]$ to denote the RBC instance where j is the sender. If not enough qualified dealers have been identified, reliably broadcast the message $\perp$.

– **Any round:** In any round during the protocol, if RBC$[j]$ has output (`qualified-set`, D_j) such that D_j is a subset of $[n]$ containing at least $\lfloor n/2 \rfloor + 1$ nodes, and moreover every node in D_j has become qualified w.r.t. node i's view so far, then node i considers j as a *candidate*, and node i records the tuple (j, D_j).

– **Round $T_{\text{vss}} + T_{\text{rbc}}$:** In round $T_{\text{vss}} + T_{\text{rbc}}$, do the following:

- invoke the Reconstruct sub-protocol of all VSS instances;
- if at least $\lfloor n/2 \rfloor + 1$ nodes are now considered candidates: let S be the set of all candidates so far; now multicast (`candidate-set`, S) along with a signature on the message.

– **Any round:** At any time, if a node i has observed a (`candidate-set`, S_j) message with a valid signature from the purported sender j where $S_j \subseteq [n]$ is at least $\lfloor n/2 \rfloor + 1$ in size, and moreover, every node in S_j is now considered a candidate by node i too, we say that node i becomes *happy* with j.

how to realize BA under 0.5-weak synchrony ?

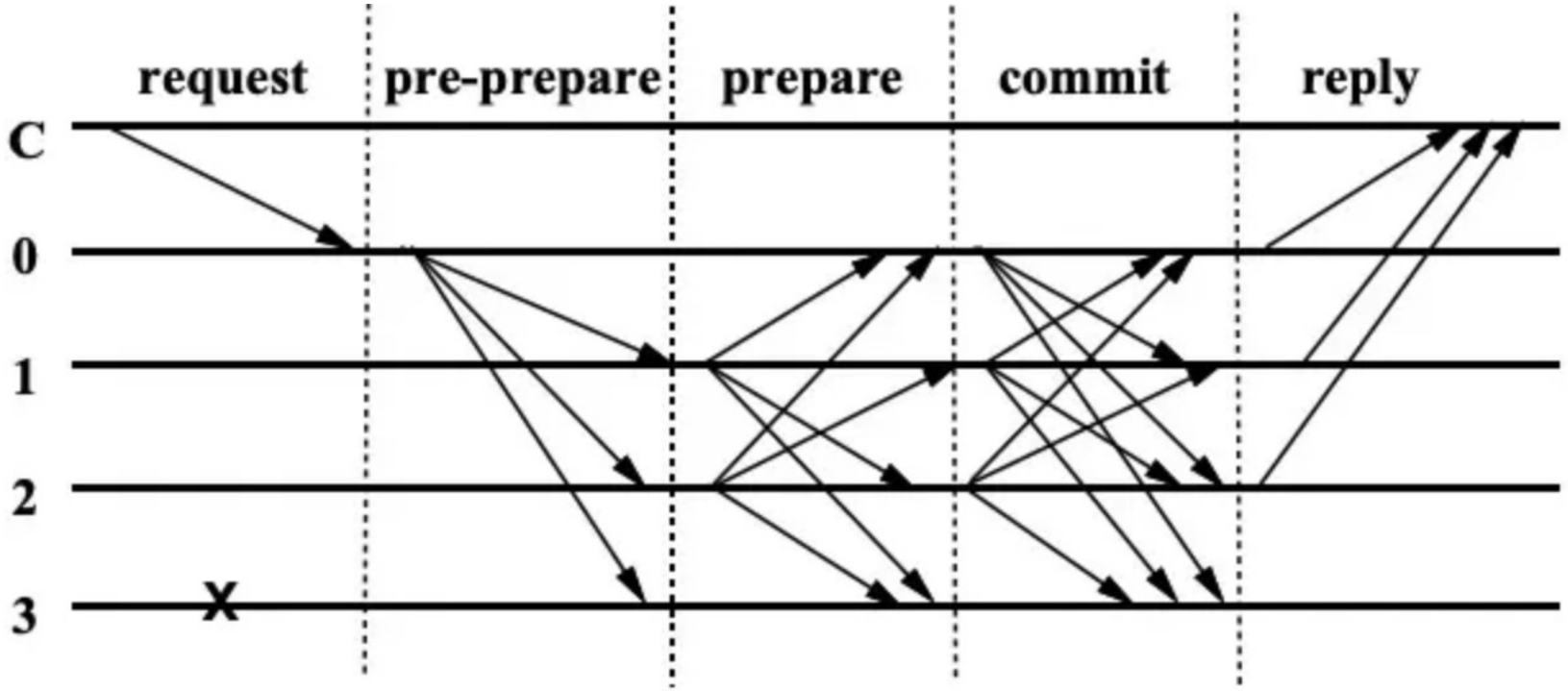
领导人选举（LE）

- **As soon as** node i becomes happy with at least $\lfloor n/2 \rfloor + 1$ nodes, let S_i^* be the current set of nodes that are considered candidates;
- **As soon as** the relevant VSS instances (needed in the following computation) have terminated the reconstruction phase outputting a reconstructed secret—henceforth let $c'_{u,v}$ be the secret reconstructed from instance $\mathsf{VSS}[u, v]$:

- For every $u \in S_i^*$: let (u, D_u) be a previously recorded tuple when u first became a candidate; compute node u's *charisma* as $C_u := \prod_{v \in D_u} c'_{v,u}$.
- Output the node $u^* \in S_i^*$ with maximum charisma (where ordering between elements in $\mathbb{F}$ is determined using lexicographical comparisons).

how to realize BA under 0.5-weak synchrony ?

拜占庭协议



– **Propose.** For the initial T_{le} rounds in each epoch, do the following:

1. If the current epoch is $e = 1$, then in round 0 of epoch 1, the sender multicasts a signed tuple $(\texttt{propose}, b)$ where b is its input bit.
2. Round 0 of every epoch: invoke an instance of the LE protocol.
3. Round T'_{le} of every epoch: every node $i \in [n]$ flips a random coin $b_i \leftarrow_{\$} \{0, 1\}$, and multicasts a signed tuple $(\texttt{propose}, b_i)$

how to realize BA under 0.5-weak synchrony ?

拜占庭协议

– **Prepare (round $T_{\text{le}} + \Delta$ of each epoch).** If $e = 1$ and a node has heard an epoch-1 proposal for b from the sender, then it multicasts the signed tuple $(\texttt{prepare}, e, b)$. Else if $e > 1$, every node performs the following:

1. if an epoch-e proposal of the form $(\texttt{propose}, e, b)$ has been heard from an eligible epoch-e proposer which is defined by the output of LE and moreover, either an epoch-$(e-1)$ commit evidence vouching for b or $\lfloor n/2 \rfloor + 1$ epoch-$(e-1)$ complaints from distinct nodes have been observed, multicast the signed tuple $(\texttt{prepare}, e, b)$.
   If LE has not produced an output in the range $[n]$ at the beginning of this round, act as if no valid proposal has been received.
2. else multicast the signed tuple $(\texttt{prepare}, e, b)$ if the node has seen an epoch-$(e-1)$ commit evidence vouching for the bit b (if both bits satisfy this then send a prepare message for each bit).

– **Commit (round $T_{\text{le}} + 2\Delta$ of each epoch).** If by the beginning of the commit round of the current epoch e, a node

1. has heard an epoch-e commit evidence for the bit b;
2. has not observed a valid epoch-e proposal for $1 - b$ (from an eligible proposer); and
3. has not observed any epoch-$(e-1)$ commit evidence for $1 - b$;

then multicast the signed tuple $(\texttt{commit}, e, b)$.

how to realize BA under 0.5-weak synchrony ?

拜占庭协议

– **Complain (round $T_{\text{le}} + 3\Delta$ of each epoch).** If no epoch-e commit evidence has been seen, multicast the signed tuple $(\texttt{complain}, e)$.
– End of this epoch and beginning of next epoch (round $T_{\text{le}} + 4\Delta$).

THANKS