# Terminating Reliable Broadcast

“终止可靠广播（Terminating Reliable Broadcast, TRB）”是一种通信原语，它在分布式系统中用于以可靠的方式在一组进程之间传播消息。这种机制在某些方面类似于可靠广播（包括统一可靠广播），但也有其独特的特点和更强的保证。

*与可靠广播的相似之处：*

1. **可靠性**：在TRB中，正确的进程必须接收到并传递所有被任何进程接收到的消息，这与可靠广播的基本要求相似。

*与可靠广播的不同之处：*

1. **指定广播源**：TRB定义了一个特定的广播源（source），该源负责广播消息。这与拜占庭一致性广播和拜占庭可靠广播类似，其中广播源事先已知并且对所有进程可见。
2. **终止性**：在TRB中，如果广播源是正确的，则所有正确的进程都必须传递广播源发出的消息。这使得TRB比普通的（统一）可靠广播更强，因为它保证了即使在广播源崩溃的情况下，每个正确的进程也必须传递一条消息。

*特殊消息△：*

- **源失败标记**：TRB使用特殊的消息`△`来表示源的失败（"Source Failure"）。如果广播源崩溃，其他进程可能会传递这个特殊的消息`△`。
- **消息域**：在TRB中，`△`不属于正常消息集`M`，用于区分普通消息和表示源故障的情况。

*操作机制：*

- **广播源**：过程`src`（即广播源）负责广播消息`m ∈ M`，其中`M`是消息集，且`△ ∉ M`。
- **消息传递**：如果广播源是正确的，其他所有正确的进程需要传递广播源发送的消息`m`。如果广播源崩溃，它们可能会传递特殊的消息`△`。

终止可靠广播是在分布式系统中传播消息的重要机制，特别是在需要确保消息传递即使在广播源失败的情况下也能完成时。

# 属性

1. **有效性（Validity）**：
   - 如果一个正确的进程`src`广播了消息`m`，那么`src`最终会传递消息`m`。这保证了由正确的源进程发出的消息将被该源进程本身传递。
2. **终止（Termination）**：
   - 每个正确的进程最终会传递恰好一条消息。这个属性确保了所有正确的进程都会在有限的时间内完成消息的传递，无论系统状态如何。
3. **完整性（Integrity）**：
   - 如果一个正确的进程传递了某条消息`m`，那么`m`要么是由源进程`src`之前广播的，要么是特殊的符号`△`。这避免了错误或重复的消息传递，并处理了源进程可能崩溃的情况。
4. **统一一致性（Uniform Agreement）**：

- 如果任何进程传递了消息`m`，那么每个正确的进程最终都会传递消息`m`。这个属性强化了系统的一致性，确保了所有正确的进程对传递的消息达成一致。

- **广播请求**：
  - `< utrb, Broadcast | m >`：源进程`src`使用此请求来广播消息`m`。
- **传递指示**：
  - `< utrb, Deliver | src, m >`：用于指示消息`m`由源进程`src`广播或传递特殊符号`△`。

# 实现

```
Consensus-Based UTRB
Implements: UniformTerminatingReliableBroadcast, instance utrb, with sender src.
Uses: BestEffortBroadcast, instance beb; UniformConsensus, instance uc;
PerfectFailureDetector, instance P.

# UTRB实例初始化
upon event < utrb, Init > do
    proposal := ⊥;  # 初始化提案为未定义

# 当UTRB接收到广播请求时
upon event < utrb, Broadcast | m > do
    trigger < beb, Broadcast | m >;  # 触发最大努力广播，广播消息m

# 当从源src接收到最大努力广播的消息时
upon event < beb, Deliver | src, m > do
    if proposal = ⊥ then  # 如果还没有提案
        proposal := m;  # 设置提案为接收到的消息
        trigger < uc, Propose | proposal >;  # 触发统一共识，提出提案

# 当检测到进程p崩溃时
upon event < P, Crash | p > do
    if p = src ∧ proposal = ⊥ then  # 如果崩溃的进程是源进程且还没有提案
        proposal := △;  # 设置提案为特殊符号△
        trigger < uc, Propose | proposal >;  # 触发统一共识，提出提案△

# 当统一共识做出决定时
upon event < uc, Decide | decided > do
    trigger < utrb, Deliver | src, decided >;  # 触发UTRB的传递事件，传递决定的消息
```

- **初始化**：在UTRB实例初始化时，将提案变量设置为未定义（⊥）。
- **广播消息**：当源进程`src`要广播消息`m`时，它通过最大努力广播（Best Effort Broadcast, BEB）机制发送这个消息。
- **接收消息并提出提案**：当任何进程通过BEB接收到源进程`src`的消息时，如果它还没有提案，它会将这个消息作为提案，并触发统一共识（Uniform Consensus）机制。
- **处理源进程崩溃**：如果检测到源进程`src`崩溃，并且当前进程还没有提案，它会将特殊符号`△`作为提案，并触发统一共识机制。
- **决定和传递**：一旦统一共识机制做出了决定，该决定会被传递给UTRB的所有参与者。

# 正确性

1. **完整性（Integrity）**：
   - 在UTRB中，由于没有其他值被提议给共识机制，因此完整性属性直接由共识的有效性保证。这意味着如果一个正确的进程传递了某个消息，那么这个消息必定是由源进程广播的，或者是特殊符号`△`。
2. **有效性（Validity）**：
   - 由于源进程`src`是正确的，根据最大努力广播（BEB）的有效性，每个正确的进程都会接收消息`m`。同时，由于完美故障检测器（P）的强准确性，没有进程会将提案更改为`△`。因此，每个进程都会将`m`提议给共识，使得`m`成为唯一可能的决定。源进程`src`会传递消息`m`。
3. **统一一致性（Uniform Agreement）**：
   - 由共识的统一一致性保证，如果任何进程传递了消息`m`，那么每个正确的进程最终都会传递消息`m`。这确保了所有正确的进程对传递的消息达成一致。
4. **终止（Termination）**：
   - 根据BEB的无重复性和完美故障检测器的强完整性，每个进程都将准确且恰好一次向共识提出提案。由于共识的终止和完整性属性，最终所有正确的进程都会做出决定。

# Group Membership

群组成员管理（Group Membership）的概念，它在分布式系统中用于协调关于进程的加入、离开和崩溃的信息。

*群组成员管理的作用*

1. **知晓参与计算的进程**：群组成员管理使进程能够知道当前哪些进程是活动的，哪些已经离开或崩溃。
2. **协调信息**：与故障检测器提供的信息相比，群组成员管理提供的信息是协调的，即所有进程都有相同的关于哪些进程是活动或崩溃的视图。

*故障检测器与群组成员管理的区别*

- **故障检测器**：
  - 提供有关进程故障的信息。
  - 即使故障检测器是完美的（如完美故障检测器P），提供的信息也可能不是协调的，即不同的进程可能有关于其他进程状态的不同看法。
- **群组成员管理**：
  - 不仅提供有关故障的信息，还协调这些信息，确保所有进程对故障有相同的认识。
  - 进程安装（install）相同的视图（views）序列，这些视图反映了当前认为是活动或崩溃的进程集合。
  - 与完美故障检测器P类似，群组成员管理提供了准确的关于故障的知识，但与之不同的是，这些信息是协调一致的。

# 属性

1. **单调性（Monotonicity）**：

- 如果一个进程 `p` 先安装了视图 `V = (id, M)`，然后安装了视图 `V' = (id', M')`，那么 `id < id'` 且 `M ⊇ M'`。这意味着视图的标识符 `id` 随时间单调递增，且随着时间的推移，成员集合 `M` 可能会缩减。

2. **统一一致性（Uniform Agreement）**：
   - 如果某个进程安装了视图 `V = (id, M)`，而另一个进程安装了某个视图 `V' = (id, M')`，那么 `M = M'`。这保证了具有相同标识符的视图在所有进程中具有相同的成员集合。
3. **完整性（Completeness）**：
   - 如果一个进程 `p` 崩溃，那么最终每个正确的进程都会安装一个视图 `(id, M)`，使得 `p ∉ M`。这确保了所有正确的进程最终都会意识到进程 `p` 的崩溃并从其成员集合中排除 `p`。
4. **准确性（Accuracy）**：
   - 如果某个进程安装了视图 `(id, M)`，且对于某个进程 `q ∈ Π` 有 `q ∉ M`，那么 `q` 已经崩溃。这表明，只有在进程实际崩溃的情况下，它才会被从成员集合中移除。

# 实现

```
Consensus-Based Group Membership
Implements: GroupMembership, instance gm.
Uses: UniformConsensus (multiple instance); PerfectFailureDetector, instance P.

# 当群组成员管理实例初始化时
upon event < gm, Init > do
    (id, M) := (0, Π);  # 初始化视图ID为0，成员集合M为所有进程
    correct := Π;  # 初始化正确进程集合为所有进程
    wait := FALSE;  # 初始化等待状态为FALSE
    trigger < gm, View | (id, M ) >;  # 触发新视图的安装

# 当检测到进程崩溃时
upon event < P, Crash | p > do
    correct := correct \ {p};  # 从正确进程集合中移除崩溃的进程

# 当正确的进程集合小于当前成员集合且不在等待状态时
upon correct ⊊ M ∧ wait = FALSE do
    id := id + 1;  # 视图ID递增
    wait := TRUE;  # 设置等待状态为TRUE
    Initialize a new instance uc.id;  # 初始化一个新的统一共识实例
    trigger < uc.id, Propose | correct >;  # 在统一共识实例上提议当前正确的进程集合

# 当统一共识做出决定时
upon event < uc.i, Decide | M' > such that i = id do
    M := M';  # 更新成员集合
    wait := FALSE;  # 设置等待状态为FALSE
    trigger < gm, View | (id, M) >;  # 触发新视图的安装
```

# 正确性

1. **单调性（Monotonicity）**：

- 假设 `id' ≤ id`，即某个统一共识实例 `id'` 在 `id` 之后决定。这在实践中是不可能的，因为进程不会在所有较低编号的实例都决定之前向 `id` 实例提出提案（由 `wait` 标志控制）。另外，由于统一共识的有效性，只有当 `correct ⊆ M` 时才会触发共识，因此可以保证 `M' ⊆ M`。

2. **统一一致性（Uniform Agreement）**：
   - 由统一共识的统一一致性保证，如果某个进程安装了视图 `V = (id, M)`，则其他所有进程最终也会安装相同的视图。
3. **完整性（Completeness）**：
   - 根据完美故障检测器（P）的强完整性，所有正确的进程最终都会怀疑进程 `p`。一旦所有正确的进程都怀疑了 `p`，不包含 `p` 的成员集合将被所有进程提议。根据统一共识的有效性，最终会安装一个不包含 `p` 的视图。
4. **准确性（Accuracy）**：
   - 由于完美故障检测器的强准确性，如果某个进程安装了不包含进程 `q` 的视图 `(id, M)`，则可以确信 `q` 已经崩溃。

# Non-Blocking Atomic Commit

非阻塞原子提交（Non-blocking Atomic Commit, NBAC）是一个在数据库事务中使用的一致性协议，它是一个特殊类型的共识问题。NBAC 确保在分布式环境中的多个参与者之间，事务要么完全提交（成功执行），要么完全中止（放弃执行），从而保证了事务的原子性。

**事务（Transactions）**

- 事务是描述对共享和分布式信息序列访问的原子程序。
- 一个事务可以通过提交（commit）或中止（abort）来结束。
- 例如，一个银行事务可能包括从 Alice 的账户中提款1000元，并存入 Bob 的账户。另一个例子可能是预订从上海到北京的东方航空航班和从北京到上海的国航航班。
- 事务的关键在于，要么两个操作都执行（事务提交），要么两个操作都不执行（事务中止）。

**事务可能中止的原因**

- 并发控制。
- 故障。

**ACID属性**

- **原子性（Atomicity）**：事务要么完全执行，要么完全不执行。
- **一致性（Consistency）**：事务将系统从一个一致的状态转变为另一个一致的状态。
- **隔离性（Isolation）**：事务的执行似乎是隔离的。
- **持久性（Durability）**：一旦事务提交，其效果是永久性的。

**非阻塞原子提交（NBAC）**

- NBAC 是二元共识的一个变种，用于达成关于事务是中止还是提交的决定。
- 与二元共识类似，每个进程有一个初始值0（no，中止）或1（yes，提交），并必须决定最终值0（中止）或1（提交）。

- 与普通共识不同的是，NBAC 中的进程寻求决定1（提交），但每个进程都有否决权，即有偏向于决定0（中止）的倾向。

# 属性NBAC

1. **终止（Termination）**：
   - 每个正确的进程最终都会做出某个决定。这保证了在分布式系统中，无论发生什么情况，每个进程最终都会知道事务是提交还是中止。
2. **中止有效性（Abort-Validity）**：
   - 进程只能在以下情况下决定中止（ABORT）：
     - 至少有一个进程提出了中止（ABORT）。
     - 或者至少有一个进程崩溃。
   - 这确保了只有在存在潜在问题时，才会中止事务。
3. **提交有效性（Commit-Validity）**：
   - 进程只有在没有任何进程提议中止（ABORT）时才能决定提交（COMMIT）。这意味着只有在所有参与事务的进程都同意时，才会提交事务。
4. **完整性（Integrity）**：
   - 没有进程会做出两次决定。这保证了每个进程对同一事务的决定是一次性的，避免了重复处理。
5. **统一一致性（Uniform Agreement）**：
   - 没有两个进程会对同一事务做出不同的决定。这意味着所有正确的进程都会就事务是否提交达成一致，从而保证了系统的一致性。

# 实现

```
Consensus-based NABC
Implements: NonBlockingAtomicCommit, instance nbac.
Uses: BestEffortBroadcast, instance beb; UniformConsensus, instance uc;
PerfectFailureDetector, instance P.

# 当 NBAC 实例初始化时
upon event < nbac, Init > do
    voted := ∅;  # 初始化已投票的进程集合为空
    proposed := FALSE;  # 初始化提议状态为 FALSE

# 当检测到进程崩溃时
upon event < P, Crash | p > do
    if proposed = FALSE then
        trigger < uc, Propose | ABORT >;  # 如果还未提议，则提议 ABORT
        proposed := TRUE;  # 设置提议状态为 TRUE

# 当有提议时（提交或中止）
upon event < nbac, Propose | v > do
    trigger < beb, Broadcast | v >;  # 通过尽力而为广播发送提议

# 当收到提议时
```

```
upon event < beb, Deliver | p, v > do
    if v = ABORT ∧ proposed = FALSE then
        trigger < uc, Propose | ABORT >;  # 如果收到中止提议且还未提议，则提议 ABORT
        proposed := TRUE;  # 设置提议状态为 TRUE
    else
        voted := voted ∪ {p};  # 将进程 p 添加到已投票集合中
        if voted = Π ∧ proposed = FALSE then
            trigger < uc, Propose | COMMIT >;  # 如果所有进程都已投票且还未提议，则提议
COMMIT
            proposed := TRUE;  # 设置提议状态为 TRUE

# 当统一共识做出决定时
upon event < uc, Decide | decided > do
    trigger < nbac, Decide | decided >;  # 触发决定事件，传递共识决定
```

- **初始化**：在NBAC实例初始化时，设置已投票的进程集合为空，提议状态为未提议。
- **处理进程崩溃**：如果检测到进程崩溃且尚未提议，则提议中止事务。
- **处理提议**：如果接收到来自其他进程的提议，根据提议的类型（提交或中止）和当前的提议状态，决定是否提议中止。
- **投票和提议**：跟踪收到的提议，如果所有进程都已经作出了提议且当前进程尚未提议，则提议提交。
- **共识决定**：一旦通过统一共识达成决定（提交或中止），触发NBAC的决定事件，并传递共识的决定。

## 正确性

1. **协议一致性（Agreement）**：
    - 由于统一共识（Uniform Consensus, ucons）机制的协议一致性保证，NBAC 也继承了这个特性。这意味着所有正确的进程都会就是否提交事务达成一致的决定。
2. **终止（Termination）**：
    - 该协议保证每个正确的进程最终都会做出决定。这是通过以下两种情况实现的：
        - 如果有进程崩溃，则根据完美故障检测器（Perfect Failure Detector, P）的强完整性，所有正确的进程最终都会向统一共识提出提议。
        - 如果没有进程崩溃，那么根据尽力而为广播（Best Effort Broadcast, beb）的有效性，所有正确的进程也会向统一共识提出提议。
    - 统一共识的终止特性确保了每个正确的进程最终都会做出决定。
3. **提交有效性（Commit Validity）**：
    - 如果决定是提交（1），那么只有在所有进程都提议提交时才可能发生。这是因为只有在所有进程都提议提交时，1 才会被提议给统一共识。
4. **中止有效性（Abort Validity）**：
    - 如果决定是中止（0），那么这只可能发生在至少有一个进程失败或提议中止的情况下。
    - 用反证法可以证明，如果所有进程都正确且都提议提交（1），但某个进程决定了中止（0），这将违反统一共识的有效性，因为根据统一共识，只有被提议的值才能被决定。