Problem

The system should support running a distributed DCR graph between multiple parties, under the assumption that any number of those parties will attempt to forge, collude, perjure or otherwise act maliciously, i. e. exhibiting byzantine failures. These distributed DCR graphs should always be accessible by all parties, connectivity permitting, and transparency must be ensured for all parties, in that they should always be aware of whether or not activities controlled by that party can be executed at any given time.

Definitions

Workflow: $G = (V, E)$
Activities: $V = \{v_1, \ \ldots, v_n\}$
Activity States: $S = \{s_1, \ \ldots, s_n\}$, where $s_i \subseteq \{Executed, Pending, Included\}$ corresponding to V_i.
Relation: $R =$ (condition, response, milestone, include, exclude), $e = (v_x, v_y, r_i)$, $E = \{e_1, \ \ldots, e_m\}$
Processes: $P = \{p_1, \ \ldots, p_o\}$.
History, $H_p(t)$: Sequence of activity executions up to, and at time t, as perceived by given process.
Execution: (v_i, t, p), where v_i is the activity executed at time t, executed by process p.
Dependant activities: An activity, v_1 is said to be dependant on another activity, v_2, when the DCR rules allowing v_1 to be executed, depends on the state of v_2.

Requirements

Consensus :

- **Termination** Eventually each correct process sets its decision value (single activity state)
- **Partial agreement** Bottom is allowed as substitute for any activity state, unless that activity is owned by process or dependant on one such process.
- **Integrity** If p_i is correct, then all correct processes decide on v_i, or $\perp$ as the ith component of their vector.

Correctness : Any state transition must be permitted by DCR logic or they will be rejected by correct nodes.

Non-repudiation : v_i must be provably proposed by p_i, within the bounds specified by any applied cryptography.

Scenarios

Scenario 1 $A \longrightarrow\!\!\bullet B$

Alice must never decide $\perp$ for A. Bob must never decide $\perp$ for either A or B.

Scenario 2 $A \longrightarrow\!\!\bullet B$, $A \longrightarrow\!\!\bullet C$

Alice, Bob and Charlie must always decide the same value for A.
Problem: Prevent Alice from telling Bob the state, but not Charlie. (Reduces to FLP.)

Scenario 3 $A \longrightarrow\!\!\circledast B$, $B \longrightarrow\!\!\bullet C$

Alice and Bob must agree on the state of A, Bob and Charlie must agree on the state of B, only Charlie must agree on the value of C.

Scenario 4 $A \longrightarrow\!\!\circledast B$, $B \longrightarrow\!\!\circledast A$

Both Alice and Bob must agree on both A and B.
Problem: Concurrency.

Scenario 5 $A \longrightarrow\!\!+ C$, $A \longrightarrow\!\!+ D$, $B \longrightarrow\!\!\circledast C$, $B \longrightarrow\!\!\circledast D$

Both Alice and Bob must agree on only A and B, respectively. Charlie must agree on everything except D, and Dahlia must agree on everything except C
Problem: Concurrency (expanded), not serially equivalent if C is included and D is excluded after A and B have executed concurrently.

Locking argument

1: Exclude-include $A \longrightarrow\!\!\circledast D, A \longrightarrow\!\!+ C, B \longrightarrow\!\!\circledast C, B \longrightarrow\!\!+ D$

2: Include condition $A \longrightarrow\!\!+ C, C \longrightarrow\!\!\bullet B, B \longrightarrow\!\!+ D, D \longrightarrow\!\!\bullet A$ where C and D are excluded.

3: Non-modifying to condition $A \longrightarrow\!\!+ C, C \longrightarrow\!\!\bullet B, B \longrightarrow\!\!+ D, D \longrightarrow\!\!\bullet A$ where D is excluded.

There are three sufficient ways of locking when executing:

2 degrees back has the disadvantage that notifications have to be pushed independently of locking. This locking does not work for graph 1, as A and B do not need to acquire locks on C and D. This means that the end state is subject to race conditions.

All adjacent notifications need to be pushed in the second degree ocasionally (graph 2).

2 degrees forward has the inherent advantage of notifying relevant activities on execution.

1 The algorithm

Consider the events A, B and C, where A has relation AB to B, and B has relation BC to C.
When A is executed:
B must be locked if:
AB is an effect (include, exclude, response)
C must be locked if:
AB is a constraining effect, and BC is a constraint (condition, milestone), in the following configuration:

- $A \rightarrow\!\!+ B \rightarrow\!\!\bullet C$ (B is excluded)
- $A \rightarrow\!\!+ B \rightarrow\!\!\diamond C$ (B is excluded and pending)
- $A \bullet\!\!\rightarrow B \rightarrow\!\!\diamond C$ (B is not pending)