Problem

Ensure that two parties collaborating on a DCR graph, can execute the activities in the graph without requiring any degree of trust. We want to ensure fairness in the sense that each collaborator will be able to execute their relevant activities (in so far as they can do so according to the specific DCR graph), and that no collaborator can repudiate executions of activities (neither their own nor other parties'). The resulting global state of the distributed executions must be a result of a valid execution sequence in the workflow.

Failure model: Byzantine failures with subsystems exhibiting only crashes (no key guessing)

1. There must be a happened-before relationship between the execution of an activity and the executions of its dependent activities 2. All serializations of the happened-before relationships must be valid execution sequences in DCR

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=$(\mathrm{v}_x, v_y, r_i)$, $E = \{e_1, \ \ldots, e_m\}$
Actors: $P = \{p_1, \ \ldots, p_o\}$.
History, $H_p(t)$: Sequence of activity executions up to, and at time t, as perceived by given actor.
Execution: (v_i, t, p), where v_i is the activity executed at time t, executed by actor p.
Dependant activities: ...
Processor: ...
Activity responsibility: ...

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 $A \rightarrow\bullet B$

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

Scenario 2 $A \rightarrow\bullet B$, $A \rightarrow\bullet C$

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 $A \rightarrow\div B$, $B \rightarrow\bullet C$

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 $A \rightarrow\div B$, $B \rightarrow\div A$

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

Scenario 5 $A \rightarrow+ C$, $A \rightarrow+ D$, $B \rightarrow\div C$, $B \rightarrow\div D$

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.

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)

- $A \bullet\!\!\longrightarrow B \longrightarrow\!\!\diamond C$ (B is not pending)