MODULE *IscpBatchTimestamp*

Let's assume the transaction currently being built is t_i and the previous one is t_{i-1}. The following requirements apply to the timestamp $t_i.ts$ of the transaction t_i:

1. Transaction timestamps are non-decreasing function in a chain, i.e.

$$t_i.ts \geq t_{i-1}.ts.$$

2. A transaction timestamp is not smaller than the timestamps of request transactions taken as inputs in t_i, i.e.

$$\forall r \in t_i.req : t_i.ts \geq t_i.req[r].tx.ts,$$

where $t_i.req$ is a list of requests processed as inputs in the transaction t_i, $t_i.req[r]$ is a particular request and $t_i.req[r].tx$ is a transaction the request belongs to.

The initial attempt was to use the timestamp $t_i.ts$ as a median of timestamps proposed by the committee nodes accepted to participate in the transaction t_i by the ACS procedure. This approach conflicts with the rules of selecting requests for the batch (take requests that are mentioned in at least $F+1$ proposals). In this way it is possible that the median is smaller than some request transaction timestamp .

In this document we model the case, when we take maximal of the proposed timestamps excluding the F highest values. This value is close to the 66th percentile (while median is the 50th percentile). In this case all the requests selected to the batch will have timestamp lower than the batch timestamp IF THE BATCH PROPOSALS MEET THE CONDITION

$$\forall p \in batchProposals : \forall r \in p.req : p.req[r].tx.ts \leq p.ts.$$

It is possible that it can be not the case, because of the byzantine nodes. The specification bellow shows, that property (2) can be violated, in the case of byzantine node sending timestamp lower than the requests in the proposal.

The receiving node thus needs to check, if the proposals are correct. For this check it must have all the transactions received before deciding the final batch. The detected invalid batch proposals must be excluded from the following procedure. But that can decrease number of requests included into the final batch (because requests are included if mentioned in $F+1$ proposals). It is safe on the receiver side to "fix" such proposals by setting their timestamp to the maximal transaction timestamp of the requests in the proposal.

EXTENDS *Naturals*, *FiniteSets*, *TLAPS*, *FiniteSetTheorems*, *NaturalsInduction*

CONSTANT *Time* — A set of timestamps, represented as natural numbers to have $\leq$.

CONSTANT *Nodes* — A set of node identifiers.

CONSTANT *Byzantine* — A set of byzantine node identifiers.

ASSUME $ConstantAssms \triangleq$

$\quad \land IsFiniteSet(Time) \land Time \neq \{\} \land Time \subseteq Nat$

$\quad \land IsFiniteSet(Nodes) \land Nodes \neq \{\}$

$\quad \land Byzantine \subseteq Nodes$

$Requests \triangleq Time$ — Assume requests are identified by timestamps of their *TX* only.

VARIABLE *acsNodes* — Nodes decided to be part of the round by the *ACS*.

VARIABLE *npRq* — Node proposal: A set of requests.

VARIABLE *npTS* — Node proposal: Timestamp.

$vars \triangleq \langle acsNodes, npRq, npTS \rangle$

$N \triangleq Cardinality(Nodes)$

$F \triangleq$ CHOOSE $F \in 0 \,..\, N$:

$\qquad \land N \geq 3 * F + 1$ — *Byzantine* quorum assumption.

$\quad\quad \land \forall f \in 0 \,..\, N : N \geq 3 * f + 1 \Rightarrow F \geq f$ Consider maximal possible F.

$\textsc{assume}\ ByzantineAssms \triangleq F \in Nat \land N \geq 3 * F + 1 \land (N \geq 4 \Rightarrow F \geq 1)$

$FQuorums \triangleq \{q \in \textsc{subset}\ Nodes : Cardinality(q) = F\}$

$F1Quorums \triangleq \{q \in \textsc{subset}\ Nodes : Cardinality(q) = F + 1\}$

$NFQuorums \triangleq \{q \in \textsc{subset}\ Nodes : Cardinality(q) = N - F\}$

$TSQuorums \triangleq \{q \in \textsc{subset}\ Nodes : q \subseteq acsNodes \land Cardinality(q) = Cardinality(acsNodes) - F\}$

BatchRqs is a set of requests selected to the batch. Requests are selected to a batch, if they are mentioned at least in $F + 1$ proposals.

$$
\begin{aligned}
& BatchRq(rq) \triangleq \exists\, q \in F1Quorums : \\
& \quad\quad \land q \subseteq acsNodes \\
& \quad\quad \land \forall n \in q : rq \in npRq[n] \\
& BatchRqs \triangleq \{rq \in Requests : BatchRq(rq)\}
\end{aligned}
$$

$BatchTS(ts)$ is a predicate, that is true for the timestamp that should be considered as a batch timestamp. It must be maximal of the batch proposals, excluding F greatest ones.

$SubsetTS(s) \triangleq \{npTS[n] : n \in s\}$

$BatchTSx(ts) \triangleq \forall\, q \in TSQuorums :$ *TODO*: Remove

$$
\begin{aligned}
& \quad\quad \land ts \in SubsetTS(q) \\
& \quad\quad \land \forall x \in SubsetTS(q) : ts \geq x \\
& \quad\quad \land \forall x \in SubsetTS(acsNodes \setminus q) : ts \leq x \\
& BatchTS(ts) \triangleq \\
& \quad \forall\, q \in FQuorums : (\\
& \quad\quad \land q \subseteq acsNodes \\
& \quad\quad \land \forall x \in q,\ y \in acsNodes \setminus q : npTS[x] \geq npTS[y] \\
& \quad) \Rightarrow (\\
& \quad\quad \land\ ts \in SubsetTS(acsNodes \setminus q) \\
& \quad\quad \land\ \forall x \in SubsetTS(acsNodes \setminus q) : ts \geq x \\
& \quad\quad \land\ \forall x \in SubsetTS(q) : ts \leq x \\
& \quad)
\end{aligned}
$$

A batch proposal is valid, if its timestamp is not less than timestamps of all the request transactions included to the proposal.

$ProposalValid(n) \triangleq \forall\, rq \in npRq[n] : rq \leq npTS[n]$

$$
\begin{aligned}
& Init \triangleq \\
& \quad \land acsNodes \in \textsc{subset}\ Nodes \land Cardinality(acsNodes) \geq N - F \\
& \quad \land npRq \in [acsNodes \rightarrow (\textsc{subset}\ Requests) \setminus \{\{\}\}] \\
& \quad \land npTS \in [acsNodes \rightarrow Time] \\
& \quad \land \forall n \in (acsNodes \setminus Byzantine) : ProposalValid(n)
\end{aligned}
$$

Fair node proposals are valid.

$Next \triangleq \textsc{unchanged}\ vars$ Only for model checking in *TLC*.

$Spec \triangleq Init \land \Box[Next]_{vars}$

$$
\begin{aligned}
& TypeOK \triangleq \\
& \quad \land acsNodes \subseteq Nodes
\end{aligned}
$$

```
   ∧ npRq ∈ [acsNodes → SUBSET Requests]
   ∧ npTS ∈ [acsNodes → Time ∪ {0}]

Invariant ≜
  ∀ ts ∈ Time, rq ∈ BatchRqs : BatchTS(ts) ⇒ rq ≤ ts

THEOREM Spec ⇒ □TypeOK ∧ □Invariant
  PROOF OMITTED  Checked with TLC.
────────────────────────────────────────────────────────────────
LEMMA SubsetsAllCardinalities ≜
  ASSUME NEW S, IsFiniteSet(S)
  PROVE ∀ x ∈ 0 . . Cardinality(S) : ∃ q ∈ SUBSET S : Cardinality(q) = x
PROOF
⟨1⟩ DEFINE P(x) ≜ x ≤ Cardinality(S) ⇒ ∃ q ∈ SUBSET S : Cardinality(q) = x
⟨1⟩1. ∀ x ∈ Nat : P(x)
  ⟨2⟩1. P(0)BY FS_EmptySet
  ⟨2⟩2. ∀ x ∈ Nat : P(x) ⇒ P(x + 1)
    ⟨3⟩1. TAKE x ∈ Nat
    ⟨3⟩2. HAVE P(x)
    ⟨3⟩3. HAVE x + 1 ≤ Cardinality(S)
    ⟨3⟩4. PICK qx ∈ SUBSET S : Cardinality(qx) = x
          BY ⟨3⟩2, ⟨3⟩3, FS_CardinalityType
    ⟨3⟩5. PICK x1 ∈ S : x1 ∉ qx
          BY ⟨3⟩3, ⟨3⟩4
    ⟨3⟩6. WITNESS qx ∪ {x1} ∈ SUBSET S
    ⟨3⟩7. Cardinality(qx ∪ {x1}) = x + 1
          BY ⟨3⟩4, ⟨3⟩5, FS_AddElement, FS_Subset
    ⟨3⟩ QED BY ⟨3⟩7
  ⟨2⟩3. QED BY ⟨2⟩1, ⟨2⟩2, NatInduction
⟨1⟩2. QED BY ⟨1⟩1

THEOREM SpecTypeOK ≜ Spec ⇒ □TypeOK
  ⟨1⟩1. Init ⇒ TypeOKBY  DEF Init, TypeOK
  ⟨1⟩2. TypeOK ∧ [Next]_vars ⇒ TypeOK′BY  DEF vars, TypeOK, Next
  ⟨1⟩3. QED BY ⟨1⟩1, ⟨1⟩2, PTL DEF Spec

THEOREM SpecInvariant ≜ Byzantine = {} ∧ Spec  ⇒ □Invariant
  ⟨1⟩ SUFFICES ASSUME Byzantine = {}PROVE Spec ⇒ □InvariantOBVIOUS
  ⟨1⟩1. TypeOK ∧ Init ⇒ Invariant
    ⟨2⟩ SUFFICES ASSUME TypeOK, InitPROVE InvariantOBVIOUS
    ⟨2⟩ USE  DEF Invariant
    ⟨2⟩ TAKE ts ∈ Time, rq ∈ BatchRqs
    ⟨2⟩ HAVE BatchTS(ts) PROVE : rq ≤ ts
    ⟨2⟩c. ∀ q1 ∈ F1Quorums, q2 ∈ NFQuorums : q1 ∩ q2 ≠ {}
          ⟨3⟩ TAKE q1 ∈ F1Quorums, q2 ∈ NFQuorums
          ⟨3⟩1. N ∈ Nat ∧ F ∈ Nat
```

```
        BY ONLY ConstantAssms, ByzantineAssms, FS_CardinalityType DEF N, F
    ⟨3⟩2. Cardinality(q1) + Cardinality(q2) > Cardinality(Nodes)
        BY ONLY ⟨3⟩1  DEF N, F1Quorums, NFQuorums
    ⟨3⟩3. q1 ⊆ Nodes ∧ q2 ⊆ Nodes
        BY ONLY  DEF F1Quorums, NFQuorums
    ⟨3⟩4. QED BY ONLY ⟨3⟩2, ⟨3⟩3, FS_MajoritiesIntersect, ConstantAssms
⟨2⟩b. ∀ rr ∈ BatchRqs : ∃ q ∈ F1Quorums : ∀ n ∈ q : rr ∈ npRq[n]
    BY  DEF BatchRqs, BatchRq
⟨2⟩d. ∀ nn ∈ acsNodes : ProposalValid(nn)
    BY  DEF Init
⟨2⟩g. acsNodes ⊆ Nodes
    BY  DEF Init
⟨2⟩f. Cardinality(acsNodes) − F > 0
    ⟨3⟩1. Cardinality(acsNodes) ∈ NatBY ⟨2⟩g, FS_CardinalityType, FS_Subset, ConstantAssms
    ⟨3⟩2. F ∈ NatBY ByzantineAssms
    ⟨3⟩3. N ∈ NatBY ConstantAssms, FS_CardinalityType DEF N
    ⟨3⟩4. Cardinality(acsNodes) ≥ N − FBY  DEF Init
    ⟨3⟩5. N − F ≥ 2 ∗ F + 1BY ByzantineAssms, ⟨3⟩2, ⟨3⟩3
    ⟨3⟩6. Cardinality(acsNodes) > FBY ⟨3⟩1, ⟨3⟩2, ⟨3⟩3, ⟨3⟩4, ⟨3⟩5, ByzantineAssms
    ⟨3⟩ QED BY ⟨3⟩1, ⟨3⟩2, ⟨3⟩6
⟨2⟩j. Cardinality(acsNodes) − F ≥ 0
    BY ⟨2⟩f
```

```
⟨2⟩i. ∃ q ∈ SUBSET Nodes : q ⊆ acsNodes ∧ Cardinality(q) = Cardinality(acsNodes) − F ⟨3⟩1.
    Cardinality(acsNodes) ∈ Nat BY ⟨2⟩g, FS_CardinalityType, FS_Subset, ConstantAssms
    ⟨3⟩6. Cardinality(Nodes) ∈ Nat BY FS_CardinalityType, ConstantAssms
    ⟨3⟩2. F ∈ Nat BY ByzantineAssms
    ⟨3⟩3. N ∈ Nat BY ConstantAssms, FS_CardinalityType DEF N ⟨3⟩4. F ≥ 0 BY
    ⟨3⟩2, FS_CardinalityType, FS_EmptySet DEF F ⟨3⟩5. F < N − F BY ⟨3⟩2, ⟨3⟩3,
    FS_CardinalityType, ConstantAssms, ByzantineAssms DEF F
    ⟨3⟩x. Cardinality(Nodes) ≥ Cardinality(acsNodes)
        BY FS_CardinalityType, FS_Subset, ConstantAssms DEF Init
    ⟨3⟩z. Cardinality(acsNodes) ≥ (Cardinality(acsNodes) − F) BY FS_CardinalityType,
        FS_EmptySet, ⟨2⟩f, ⟨3⟩1, ⟨3⟩4, ⟨3⟩5, ByzantineAssms
    ⟨3⟩y. Cardinality(Nodes) ≥ (Cardinality(acsNodes) − F) BY ONLY ⟨3⟩x, ⟨3⟩z, ⟨3⟩1, ⟨3⟩2,
        ⟨3⟩6
    ⟨3⟩w1. ∃ q ∈ SUBSET Nodes : Cardinality(q) ≥ (Cardinality(acsNodes) − F) BY ⟨3⟩y,
        FS_CardinalityType, FS_Subset, FS_SUBSET
    ⟨3⟩w2. ∃ q ∈ SUBSET Nodes : Cardinality(q) ≤ (Cardinality(acsNodes) − F) BY
        ⟨2⟩j, ⟨3⟩1, ⟨3⟩2, FS_CardinalityType, FS_Subset, FS_SUBSET, FS_EmptySet,
        ConstantAssms
    ⟨3⟩ DEFINE TSQCard ≜ Cardinality(acsNodes) − F
    ⟨3⟩u1. ∃ q ∈ SUBSET Nodes : Cardinality(q) ≥ 0 ∧ Cardinality(q) ≤ Cardinality(Nodes)BY
        FS_CardinalityType, FS_Subset, FS_SUBSET, ConstantAssms
    ⟨3⟩u2. TSQCard ≥ 0 BY ⟨2⟩j
    ⟨3⟩u3. TSQCard ≤ Cardinality(Nodes)BY ⟨3⟩y
    ⟨3⟩u4. TSQCard ∈ Nat BY ⟨3⟩1, ⟨3⟩2, ⟨2⟩f
    ⟨3⟩ HIDE DEF TSQCard
```

$\langle 3\rangle$u5. $\forall q \in$ SUBSET *Nodes* : $Cardinality(q) \in Nat$ BY *FS_Subset*, *FS_CardinalityType*, *ConstantAssms*

\ * $\langle 3\rangle$u6. $\forall x \in 0 \,..\, Cardinality(Nodes) : \exists q \in$ SUBSET *Nodes* : $Cardinality(q) = x$

\ * BY *ConstantAssms*, *FS_CardinalityType*, *FS_Subset*, *FS_SUBSET* $\langle 3\rangle$q. $\exists q \in$ SUBSET *Nodes* : $Cardinality(q) = TSQCard$

BY $\langle 3\rangle$u1, $\langle 3\rangle$u2, $\langle 3\rangle$u3, $\langle 3\rangle$u4, $\langle 3\rangle$u5, $\langle 3\rangle$6, *FS_Subset*, *FS_CardinalityType*, *ConstantAssms*, *SubsetsAllCardinalities*

$\langle 3\rangle$ PICK $q \in$ SUBSET *Nodes* : $Cardinality(q) = Cardinality(acsNodes) - F$ BY $\langle 3\rangle$q, *FS_CardinalityType*, *FS_Subset*, *FS_SUBSET* DEF *TSQCard*

$\langle 3\rangle$ QED OBVIOUS

$\langle 2\rangle$h. $TSQuorums \neq \{\}$ \ * $TSQuorums \triangleq \{q \in$ SUBSET *Nodes* : $q \subseteq acsNodes \land Cardinality(q) = Cardinality(acsNodes) - F\}$

BY $\langle 2\rangle$f DEF *TSQuorums*

$\langle 2\rangle$ QED OBVIOUS PROOF OMITTED \ * *TODO*

$\langle 1\rangle$2. $Invariant \land [Next]_{vars} \Rightarrow Invariant'$

$\langle 2\rangle$1. SUFFICES ASSUME *Invariant*PROVE $[Next]_{vars} \Rightarrow Invariant'$

OBVIOUS

$\langle 2\rangle$2. UNCHANGED $vars \Rightarrow (Invariant')$

BY $\langle 2\rangle$1 DEF *vars*, *Invariant*, *BatchRq*, *BatchRqs*, *BatchTS*, *ProposalValid*, *SubsetTS*, *TSQuorums*

$\langle 2\rangle$3. SUFFICES ASSUME *Next*PROVE $Invariant'$

BY $\langle 2\rangle$2

$\langle 2\rangle$4. QED BY $\langle 2\rangle$1, $\langle 2\rangle$3 DEF *vars*, *Next*, *Invariant*, *BatchRq*, *BatchRqs*, *BatchTS*, *ProposalValid*, *SubsetTS*, *TSQuorums*

$\langle 1\rangle$q. QED BY $\langle 1\rangle$1, $\langle 1\rangle$2, *PTL*, *SpecTypeOK* DEF *Spec*, *vars*

THEOREM $SpecInvariant \triangleq Byzantine = \{\} \land Spec \Rightarrow \Box Invariant$

$\langle 1\rangle$ SUFFICES ASSUME $Byzantine = \{\}$PROVE $Spec \Rightarrow \Box Invariant$ OBVIOUS

$\langle 1\rangle$1. $Init \Rightarrow Invariant$ BY DEF *Init*, *Invariant* $\langle 1\rangle$2. $TypeOK \land TypeOK' \land Invariant \land [Propose]\_vars \Rightarrow Invariant'$

$\langle 2\rangle$1. SUFFICES ASSUME *TypeOK*, $TypeOK'$, *Invariant* PROVE $[Propose]\_vars \Rightarrow Invariant'$

OBVIOUS

$\langle 2\rangle$2. UNCHANGED $vars \Rightarrow (Invariant')$ BY $\langle 2\rangle$1 DEF *vars*, *Invariant*, *BatchRq*, *BatchRqs*, *BatchTS*,

ProposalValid, *SubsetTS*, *TSQuorums*

$\langle 2\rangle$3. SUFFICES ASSUME *Propose* PROVE $Invariant'$

BY $\langle 2\rangle$2

$\langle 2\rangle$4. SUFFICES ASSUME $proposed'$

PROVE $(\forall ts \in Time, rq \in BatchRqs\colon BatchTS(ts) \Rightarrow rq \leq ts)'$

BY DEF *Invariant*

$\langle 2\rangle$ TAKE $ts \in Time$, $rq \in BatchRqs'$

$\langle 2\rangle$ HAVE $BatchTS(ts)'$ \ * PROVE : $rq \leq ts$

\ * $\langle 2\rangle$a1. $\forall n \in Nodes \setminus Byzantine : (\forall rqx \in npRq[n] : rqx \leq npTS[n])'$

\ * BY DEF *Propose*, *ProposalValid*

\ * $\langle 2\rangle$a2. $\forall n \in Nodes : (\forall rqx \in npRq[n] : rqx \leq npTS[n])'$

\ * BY $\langle 2\rangle$a1

$\langle 2\rangle$b. $\forall x \in BatchRqs : \exists q \in F1Quorums : \forall n \in q\colon x \in npRq[n]$BY DEF *BatchRqs*, *BatchRq*

$\langle 2\rangle$c. $\forall\, q1 \in F1Quorums,\ q2 \in NFQuorums : q1 \cap q2 \neq \{\}$
$\langle 3\rangle$ TAKE $q1 \in F1Quorums,\ q2 \in NFQuorums$
$\langle 3\rangle 1.$ $N \in Nat \land F \in Nat$
BY ONLY $ConstantAssms,\ ByzantineAssms,\ FS\_CardinalityType$ DEF $N,\ F$
$\langle 3\rangle 2.$ $Cardinality(q1) + Cardinality(q2) > Cardinality(Nodes)$
BY ONLY $\langle 3\rangle 1$ DEF $N,\ F1Quorums,\ NFQuorums$
$\langle 3\rangle 3.$ $q1 \subseteq Nodes \land q2 \subseteq Nodes$
BY ONLY DEF $F1Quorums,\ NFQuorums$
$\langle 3\rangle 4.$ QED BY ONLY $\langle 3\rangle 2,\ \langle 3\rangle 3,\ FS\_MajoritiesIntersect,\ ConstantAssms$
$\langle 2\rangle$q. QED PROOF OMITTED $\backslash * \ TODO$
$\backslash *$ BY DEF $BatchTS,\ BatchRq,\ BatchRqs,\ SubsetTS$
$\backslash *$, $Requests,\ NFQuorums,\ F1Quorums,\ N,\ F$
$\langle 1\rangle$q. QED BY $\langle 1\rangle 1,\ \langle 1\rangle 2,\ PTL,\ SpecTypeOK$ DEF $Spec,\ vars$

Counter-example with $Nodes = 101 \,..\, 104$, $Byzantine = \{104\}$, $Time = 1 \,..\, 3$:
$PropposedRq$: $(101 :> \{1\} @@ 102 :> \{1\} @@ 103 :> \{2\} @@ 104 :> \{2\})$,
$PropposedTS$: $(101 :> 1 @@ 102 :> 1 @@ 103 :> 2 @@ 104 :> 1\ )$,
$BatchRq$: $\{1,\ 2\}$,
$BatchTS$: 1