Stellar Quorum Systems

Giuliano Losa
Galois, Inc., USA
giuliano@galois.com

February 6, 2026

Abstract

We formalize the static properties of personal Byzantine quorum systems (PBQSs) and Stellar quorum systems, as described by Losa, Gafni, and Mazières in "Stellar Consensus by Instantiation", 33rd International Symposium on Distributed Computing (DISC 2019).

Contents

This theory formalizes some of the results appearing in the paper "Stellar Consensus By Instantiation"[1]. We prove static properties of personal Byzantine quorum systems and Stellar quorum systems.

```
theory Stellar-Quorums
  imports Main
begin
```

1 Personal Byzantine quorum systems

```
locale personal-quorums =
  fixes quorum-of :: 'node ⇒ 'node set ⇒ bool
  assumes quorum-assm:⋀ p p' . ⟦quorum-of p Q; p' ∈ Q⟧ ⟹ quorum-of p' Q
   — In other words, a quorum (of some participant) is a quorum of all its members.
begin

definition blocks where
  — Set R blocks participant p.
  blocks R p ≡ ∀ Q . quorum-of p Q ⟶ Q ∩ R ≠ {}

abbreviation blocked-by where blocked-by R ≡ {p . blocks R p}

lemma blocked-blocked-subset-blocked:
  blocked-by (blocked-by R) ⊆ blocked-by R
proof −
  have False if p ∈ blocked-by (blocked-by R) and p ∉ blocked-by R for p
  proof −
    have Q ∩ blocked-by R ≠ {} if quorum-of p Q for Q
      using ‹p ∈ blocked-by (blocked-by R)› that unfolding blocks-def by auto
    have Q ∩ R ≠ {} if  quorum-of p Q for Q
    proof −
      obtain p' where p' ∈ blocked-by R and p' ∈ Q
          by (meson Int-emptyI ‹⋀Q. quorum-of p Q ⟹ Q ∩ blocked-by R ≠ {}›
‹quorum-of p Q›)
      hence quorum-of p' Q
        using quorum-assm that by blast
      with ‹p' ∈ blocked-by R› show Q ∩ R ≠ {}
        using blocks-def by auto
    qed
    hence p ∈ blocked-by R by (simp add: blocks-def)
    thus False using that(2) by auto
  qed
  thus blocked-by (blocked-by R) ⊆ blocked-by R
    by blast
qed

end
```

We now add the set of correct participants to the model.

locale *with-w* = *personal-quorums quorum-of* **for** *quorum-of* :: $'node \Rightarrow 'node\ set \Rightarrow bool$ +
fixes W::$'node\ set$ — W is the set of correct participants
begin

abbreviation B **where** $B \equiv -W$
— B is the set of malicious participants.

definition *quorum-of-set* **where** *quorum-of-set* $S\ Q \equiv \exists\ p \in S$. *quorum-of* $p\ Q$

1.1 The set of participants not blocked by malicious participants

definition L **where** $L \equiv W - (blocked\text{-}by\ B)$

lemma *l2*: $p \in L \Longrightarrow \exists\ Q \subseteq W$. *quorum-of* $p\ Q$
unfolding *L-def blocks-def* **using** *DiffD2* **by** *auto*

lemma *l3*: — If a participant is not blocked by the malicious participants, then it has a quorum consisting exclusively of correct participants which are not blocked by the malicious participants.
assumes $p \in L$ **shows** $\exists\ Q \subseteq L$. *quorum-of* $p\ Q$
proof −
have *False* **if** $\bigwedge Q$. *quorum-of* $p\ Q \Longrightarrow Q \cap (-L) \neq \{\}$
proof −
obtain Q **where** *quorum-of* $p\ Q$ **and** $Q \subseteq W$
using *l2* ‹$p \in L$› **by** *auto*
have $Q \cap (-L) \neq \{\}$ **using** *that* ‹*quorum-of* $p\ Q$› **by** *simp*
obtain p' **where** $p' \in Q \cap (-L)$ **and** *quorum-of* $p'\ Q$
using ‹$Q \cap - L \neq \{\}$› ‹*quorum-of* $p\ Q$› *inf.left-idem quorum-assm* **by** *fastforce*

hence $Q \cap B \neq \{\}$ **unfolding** *L-def*
using *CollectD Compl-Diff-eq Int-iff inf-le1 personal-quorums.blocks-def personal-quorums-axioms subset-empty* **by** *fastforce*
thus *False* **using** ‹$Q \subseteq W$› **by** *auto*
qed
thus *?thesis* **by** (*metis disjoint-eq-subset-Compl double-complement*)
qed

1.2 Consensus clusters and intact sets

definition *is-intertwined* **where**
— This definition is not used in this theory, but we include it to formalize the notion of intertwined set appearing in the DISC paper.
is-intertwined $S \equiv S \subseteq W$
$\wedge$ ($\forall\ Q\ Q'$. *quorum-of-set* $S\ Q \wedge$ *quorum-of-set* $S\ Q' \longrightarrow W \cap Q \cap Q' \neq \{\}$)

definition *is-cons-cluster* **where**
— Consensus clusters

is-cons-cluster $C \equiv C \subseteq W \wedge (\forall\ p \in C\ .\ \exists\ Q \subseteq C\ .\ \textit{quorum-of}\ p\ Q)$
$\wedge\ (\forall\ Q\ Q'\ .\ \textit{quorum-of-set}\ C\ Q \wedge \textit{quorum-of-set}\ C\ Q' \longrightarrow W \cap Q \cap Q' \neq \{\})$

definition *strong-consensus-cluster* **where**
strong-consensus-cluster $I \equiv I \subseteq W \wedge (\forall\ p \in I\ .\ \exists\ Q \subseteq I\ .\ \textit{quorum-of}\ p\ Q)$
$\wedge\ (\forall\ Q\ Q'\ .\ \textit{quorum-of-set}\ I\ Q \wedge \textit{quorum-of-set}\ I\ Q' \longrightarrow I \cap Q \cap Q' \neq \{\})$

lemma *strong-consensus-cluster-imp-cons-cluster*:
— Every intact set is a consensus cluster
shows *strong-consensus-cluster* $I \Longrightarrow$ *is-cons-cluster* I
unfolding *strong-consensus-cluster-def is-cons-cluster-def*
by *blast*

lemma *cons-cluster-neq-cons-cluster*:
— Some consensus clusters are not strong consensus clusters and have no superset that is a strong consensus cluster.
shows *is-cons-cluster* $I \wedge (\forall\ J\ .\ I \subseteq J \longrightarrow \neg\textit{strong-consensus-cluster}\ J)$ **nitpick**[*falsify=false*, *card 'node=3*, *expect=genuine*]
oops

Next we show that the union of two consensus clusters that intersect is a consensus cluster.

theorem *cluster-union*:
assumes *is-cons-cluster* C_1 **and** *is-cons-cluster* C_2 **and** $C_1 \cap C_2 \neq \{\}$
shows *is-cons-cluster* $(C_1 \cup C_2)$
proof −
have $C_1 \cup C_2 \subseteq W$
using *assms*(*1*) *assms*(*2*) *is-cons-cluster-def* **by** *auto*
moreover
have $\forall\ p \in (C_1 \cup C_2)\ .\ \exists\ Q \subseteq (C_1 \cup C_2)\ .\ \textit{quorum-of}\ p\ Q$
using ‹*is-cons-cluster* C_1› ‹*is-cons-cluster* C_2› **unfolding** *is-cons-cluster-def*
by (*meson UnE le-supI1 le-supI2*)
moreover
have $W \cap Q_1 \cap Q_2 \neq \{\}$
if *quorum-of-set* $(C_1 \cup C_2)\ Q_1$ **and** *quorum-of-set* $(C_1 \cup C_2)\ Q_2$
for $Q_1\ Q_2$
proof −
have $W \cap Q_1 \cap Q_2 \neq \{\}$ **if** *quorum-of-set* $C\ Q_1$ **and** *quorum-of-set* $C\ Q_2$ **and** $C = C_1 \vee C = C_2$ **for** C
using ‹*is-cons-cluster* C_1› ‹*is-cons-cluster* C_2› ‹*quorum-of-set* $(C_1 \cup C_2)\ Q_1$› ‹*quorum-of-set* $(C_1 \cup C_2)\ Q_2$› *that*
unfolding *quorum-of-set-def is-cons-cluster-def* **by** *metis*
moreover
have ‹$W \cap Q_1 \cap Q_2 \neq \{\}$› **if** *is-cons-cluster* C_1 **and** *is-cons-cluster* C_2
and $C_1 \cap C_2 \neq \{\}$ **and** *quorum-of-set* $C_1\ Q_1$ **and** *quorum-of-set* $C_2\ Q_2$
for $C_1\ C_2$ — We generalize to avoid repeating the argument twice
proof −
obtain $p\ Q$ **where** *quorum-of* $p\ Q$ **and** $p \in C_1 \cap C_2$ **and** $Q \subseteq C_2$

using ‹C_1 ∩ C_2 ≠ {}› ‹*is-cons-cluster* C_2› **unfolding** *is-cons-cluster-def* **by** *blast*
have $Q \cap Q_1 \neq \{\}$ **using** ‹*is-cons-cluster* C_1› ‹*quorum-of-set* C_1 Q_1› ‹*quorum-of p Q*› ‹$p \in C_1 \cap C_2$›
unfolding *is-cons-cluster-def quorum-of-set-def*
by (*metis Int-assoc Int-iff inf-bot-right*)
hence *quorum-of-set* C_2 Q_1 **using** ‹$Q \subseteq C_2$› ‹*quorum-of-set* C_1 Q_1› *quorum-assm* **unfolding** *quorum-of-set-def* **by** *blast*
thus $W \cap Q_1 \cap Q_2 \neq \{\}$ **using** ‹*is-cons-cluster* C_2› ‹*quorum-of-set* C_2 Q_2›
unfolding *is-cons-cluster-def* **by** *blast*
qed
ultimately show *?thesis* **using** *assms that* **unfolding** *quorum-of-set-def* **by** *auto*
qed
ultimately show *?thesis* **using** *assms*
unfolding *is-cons-cluster-def* **by** *simp*
qed

Similarly, the union of two strong consensus clusters is a strong consensus cluster.

lemma *strong-cluster-union*:
assumes *strong-consensus-cluster* C_1 **and** *strong-consensus-cluster* C_2 **and** $C_1 \cap C_2 \neq \{\}$
shows *strong-consensus-cluster* $(C_1 \cup C_2)$
proof −
have $C_1 \cup C_2 \subseteq W$
using *assms*(*1*) *assms*(*2*) *strong-consensus-cluster-def* **by** *auto*
moreover
have $\forall\ p \in (C_1 \cup C_2)\ .\ \exists\ Q \subseteq (C_1 \cup C_2)\ .$ *quorum-of p Q*
using ‹*strong-consensus-cluster* C_1› ‹*strong-consensus-cluster* C_2› **unfolding** *strong-consensus-cluster-def*
by (*meson UnE le-supI1 le-supI2*)
moreover
have $(C_1 \cup C_2) \cap Q_1 \cap Q_2 \neq \{\}$
if *quorum-of-set* $(C_1 \cup C_2)$ Q_1 **and** *quorum-of-set* $(C_1 \cup C_2)$ Q_2
for Q_1 Q_2
proof −
have $C \cap Q_1 \cap Q_2 \neq \{\}$ **if** *quorum-of-set* C Q_1 **and** *quorum-of-set* C Q_2 **and** $C = C_1 \lor C = C_2$ **for** C
using ‹*strong-consensus-cluster* C_1› ‹*strong-consensus-cluster* C_2› *that*
unfolding *quorum-of-set-def strong-consensus-cluster-def* **by** *metis*
hence $(C_1 \cup C_2) \cap Q_1 \cap Q_2 \neq \{\}$ **if** *quorum-of-set* C Q_1 **and** *quorum-of-set* C Q_2 **and** $C = C_1 \lor C = C_2$ **for** C
by (*metis Int-Un-distrib2 disjoint-eq-subset-Compl sup.boundedE that*)
moreover
have ‹$(C_1 \cup C_2) \cap Q_1 \cap Q_2 \neq \{\}$› **if** *strong-consensus-cluster* C_1 **and** *strong-consensus-cluster* C_2
and $C_1 \cap C_2 \neq \{\}$ **and** *quorum-of-set* C_1 Q_1 **and** *quorum-of-set* C_2 Q_2
for C_1 C_2 — We generalize to avoid repeating the argument twice

proof −
obtain *p Q* **where** *quorum-of p Q* **and** $p \in C_1 \cap C_2$ **and** $Q \subseteq C_2$
using ‹$C_1 \cap C_2 \neq \{\}$› ‹*strong-consensus-cluster* C_2› **unfolding** *strong-consensus-cluster-def* **by** *blast*
have $Q \cap Q_1 \neq \{\}$ **using** ‹*strong-consensus-cluster* C_1› ‹*quorum-of-set* C_1 Q_1› ‹*quorum-of p Q*› ‹$p \in C_1 \cap C_2$›
unfolding *strong-consensus-cluster-def quorum-of-set-def*
by (*metis Int-assoc Int-iff inf-bot-right*)
hence *quorum-of-set* C_2 Q_1 **using** ‹$Q \subseteq C_2$› ‹*quorum-of-set* C_1 Q_1› *quorum-assm* **unfolding** *quorum-of-set-def* **by** *blast*
thus $(C_1 \cup C_2) \cap Q_1 \cap Q_2 \neq \{\}$ **using** ‹*strong-consensus-cluster* C_2› ‹*quorum-of-set* C_2 Q_2›
unfolding *strong-consensus-cluster-def* **by** *blast*
qed
ultimately show *?thesis* **using** *assms that* **unfolding** *quorum-of-set-def* **by** *auto*
qed
ultimately show *?thesis* **using** *assms*
unfolding *strong-consensus-cluster-def* **by** *simp*
qed

end

2 Stellar quorum systems

locale *stellar* =
fixes *slices* :: *'node ⇒ 'node set set* — the quorum slices
and *W* :: *'node set* — the well-behaved nodes
assumes *slices-ne*:$\bigwedge p\ .\ p \in W \Longrightarrow slices\ p \neq \{\}$
begin

definition *quorum* **where**
$quorum\ Q \equiv \forall\ p \in Q \cap W\ .\ (\exists\ Sl \in slices\ p\ .\ Sl \subseteq Q)$

definition *quorum-of* **where** $quorum\text{-}of\ p\ Q \equiv quorum\ Q \wedge (p \notin W \vee (\exists\ Sl \in slices\ p\ .\ Sl \subseteq Q))$
— TODO: $p \notin W$ needed?

lemma *quorum-union*:$quorum\ Q \Longrightarrow quorum\ Q' \Longrightarrow quorum\ (Q \cup Q')$
unfolding *quorum-def*
by (*metis IntE Int-iff UnE inf-sup-aci*(*1*) *sup.coboundedI1 sup.coboundedI2*)

lemma *l1*:
assumes $\bigwedge\ p\ .\ p \in S \Longrightarrow \exists\ Q \subseteq S\ .\ quorum\text{-}of\ p\ Q$ **and** $p \in S$
shows *quorum-of p S* **using** *assms* **unfolding** *quorum-of-def quorum-def*
by (*meson Int-iff subset-trans*)

lemma *is-pbqs*:
assumes *quorum-of p Q* **and** $p' \in Q$

shows *quorum-of* p' Q
— This is the property required of a PBQS.
using *assms*
by (*simp add*: *quorum-def quorum-of-def*)

interpretation *with-w quorum-of*
— Stellar quorums form a personal quorum system.
unfolding *with-w-def personal-quorums-def*
quorum-def quorum-of-def **by** *simp*

lemma *quorum-is-quorum-of-some-slice*:
assumes *quorum-of* p Q **and** $p \in W$
obtains S **where** $S \in$ *slices* p **and** $S \subseteq Q$
and $\bigwedge p' .\ p' \in S \cap W \Longrightarrow$ *quorum-of* p' Q
using *assms* **unfolding** *quorum-def quorum-of-def* **by** *fastforce*

lemma *is-cons-cluster* $C \Longrightarrow$ *quorum* C
— Every consensus cluster is a quorum.
unfolding *is-cons-cluster-def*
by (*metis inf.order-iff l1 quorum-of-def stellar.quorum-def stellar-axioms*)

2.1 Properties of blocking sets

inductive *blocking-min* **where**
— This is the set of correct participants that are eventually blocked by a set R when byzantine processors do not take steps.
$[\![p \in W;\ \forall\ Sl \in \mathit{slices}\ p\ .\ \exists\ q \in Sl{\cap}W\ .\ q \in R \vee \mathit{blocking\text{-}min}\ R\ q]\!] \Longrightarrow \mathit{blocking\text{-}min}\ R\ p$
inductive-cases *blocking-min-elim*:*blocking-min* R p

inductive *blocking-max* **where**
— This is the set of participants that are eventually blocked by a set R when byzantine processors help epidemic propagation.
$[\![p \in W;\ \forall\ Sl \in \mathit{slices}\ p\ .\ \exists\ q \in Sl\ .\ q \in R{\cup}B \vee \mathit{blocking\text{-}max}\ R\ q]\!] \Longrightarrow \mathit{blocking\text{-}max}\ R\ p$
inductive-cases *blocking-max* R p

Next we show that if R blocks p and p belongs to a consensus cluster S, then $R \cap S \neq \{\}$.

We first prove two auxiliary lemmas:

lemma *cons-cluster-wb*:$p \in C \Longrightarrow$ *is-cons-cluster* $C \Longrightarrow p \in W$
using *is-cons-cluster-def* **by** *fastforce*

lemma *cons-cluster-has-ne-slices*:
assumes *is-cons-cluster* C **and** $p \in C$
and $Sl \in$ *slices* p
shows $Sl \neq \{\}$
using *assms* **unfolding** *is-cons-cluster-def quorum-of-set-def quorum-of-def quorum-def*

by (*metis empty-iff inf-bot-left inf-bot-right subset-refl*)

lemma *cons-cluster-has-cons-cluster-slice*:
assumes *is-cons-cluster C* **and** *p*∈*C*
obtains *Sl* **where** *Sl* ∈ *slices p* **and** *Sl* ⊆ *C*
using *assms* **unfolding** *is-cons-cluster-def quorum-of-set-def quorum-of-def quorum-def*
by (*metis Int-commute empty-iff inf.order-iff inf-bot-right le-infI1*)

theorem *blocking-max-intersects-intact*:
— if R blocks p when malicious participants help epidemic propagation, and p belongs to a consensus cluster C, then $R \cap C \neq \{\}$
assumes *blocking-max R p* **and** *is-cons-cluster C* **and** $p \in C$
shows $R \cap C \neq \{\}$ **using** *assms*
proof (*induct*)
case (*1 p R*)
obtain *Sl* **where** *Sl* ∈ *slices p* **and** *Sl* ⊆ *C* **using** *cons-cluster-has-cons-cluster-slice*
using *1.prems* **by** *blast*
moreover have *Sl* ⊆ *W* **using** *assms*(*2*) *calculation*(*2*) *is-cons-cluster-def* **by** *auto*
ultimately show *?case*
using *1.hyps assms*(*2*) **by** *fastforce*
qed

Now we show that if $p \in C$, C is a consensus cluster, and quorum Q is such that $Q \cap C \neq \{\}$, then $Q \cap W$ blocks p.

We start by defining the set of participants reachable from a participant through correct participants. Their union trivially forms a quorum. Moreover, if p is not blocked by a set R, then we show that the set of participants reachable from p and not blocked by R forms a quorum disjoint from R. It follows that if p is a member of a consensus cluster C and Q is a quorum of a member of C, then $Q \cap W$ must block p, as otherwise quorum intersection would be violated.

inductive *not-blocked* **for** *p R* **where**
⟦*Sl* ∈ *slices p*; ∀ *q* ∈ *Sl*∩*W* . *q* ∉ *R* ∧ ¬*blocking-min R q*; *q* ∈ *Sl*⟧ ⟹ *not-blocked p R q*
| ⟦*not-blocked p R p′*; *p′* ∈ *W*; *Sl* ∈ *slices p′*; ∀ *q* ∈ *Sl*∩*W* . *q* ∉ *R* ∧ ¬*blocking-min R q*; *q* ∈ *Sl*⟧ ⟹ *not-blocked p R q*
inductive-cases *not-blocked-cases*:*not-blocked p R q*

lemma *l4*:
fixes *Q p R*
defines *Q* ≡ {*q* . *not-blocked p R q*}
shows *quorum Q*
proof −
have ∃ *S* ∈ *slices n* . *S* ⊆ *Q* **if** *n*∈*Q*∩*W* **for** *n*
proof−
have *not-blocked p R n* **using** *assms that* **by** *blast*

hence *n* ∉ *R* **and** ¬*blocking-min R n* **by** (*metis Int-iff not-blocked.simps that*)+
thus *?thesis* **using** *blocking-min.intros not-blocked.intros*(*2*) *that* **unfolding** *Q-def*
by (*simp*; *metis mem-Collect-eq subsetI*)
qed
thus *?thesis* **by** (*simp add*: *quorum-def*)
qed

lemma *l5*:
fixes *Q p R*
defines *Q* ≡ {*q* . *not-blocked p R q*}
assumes ¬*blocking-min R p* **and** ‹*p*∈*C*› **and** ‹*is-cons-cluster C*›
shows *quorum-of p Q*
proof −
have *p*∈*W*
using *assms*(*3*,*4*) *cons-cluster-wb* **by** *blast*
obtain *Sl* **where** *Sl* ∈ *slices p* **and** ∀ *q* ∈ *Sl*∩*W* . *q* ∉ *R* ∧ ¬*blocking-min R q*
by (*meson* ‹*p* ∈ *W*› *assms*(*2*) *blocking-min.intros*)
hence *Sl* ⊆ *Q* **unfolding** *Q-def* **using** *not-blocked.intros*(*1*) **by** *blast*
with *l4* ‹*Sl* ∈ *slices p*› **show** *quorum-of p Q*
using *Q-def quorum-of-def* **by** *blast*
qed

lemma *cons-cluster-ne-slices*:
assumes *is-cons-cluster C* **and** *p*∈*C* **and** *Sl* ∈ *slices p*
shows *Sl*≠{}
using *assms cons-cluster-has-ne-slices cons-cluster-wb stellar.quorum-of-def stellar-axioms* **by** *fastforce*

lemma *l6*:
fixes *Q p R*
defines *Q* ≡ {*q* . *not-blocked p R q*}
shows *Q* ∩ *R* ∩ *W* = {}
proof −
have *q* ∉ *R* **if** *not-blocked p R q* **and** *q*∈ *W* **for** *q* **using** *that*
by (*metis Int-iff not-blocked.simps*)
thus *?thesis* **unfolding** *Q-def* **by** *auto*
qed

theorem *quorum-blocks-cons-cluster*:
assumes *quorum-of-set C Q* **and** *p*∈*C* **and** *is-cons-cluster C*
shows *blocking-min* (*Q* ∩ *W*) *p*
proof (*rule ccontr*)
assume ¬ *blocking-min* (*Q* ∩ *W*) *p*
have *p*∈*W* **using** *assms*(*2*,*3*) *is-cons-cluster-def* **by** *auto*
define *Q′* **where** *Q′* ≡ {*q* . *not-blocked p* (*Q*∩*W*) *q*}
have *quorum-of p Q′* **using** *Q′-def* ‹¬ *blocking-min* (*Q* ∩ *W*) *p*› *assms*(*2*) *assms*(*3*) *l5*(*1*) **by** *blast*

moreover have $Q' \cap Q \cap W = \{\}$
  using *Q'-def l6* **by** *fastforce*
ultimately show *False* **using** *assms* **unfolding** *is-cons-cluster-def*
  by (*metis Int-commute inf-sup-aci*(*2*) *quorum-of-set-def*)
qed

2.2 Reachability through a set

Here we define the part of a quorum Q of p that is reachable through correct participants from p. We show that if p and p' are members of the same consensus cluster and Q is a quorum of p and Q' is a quorum of p', then the intersection $Q \cap Q' \cap W$ is reachable from both p and p' through the consensus cluster.

inductive *reachable-through* **for** p S **where**
  reachable-through p S p
| $[\![$*reachable-through* p S p'; $p' \in W$; $Sl \in$ *slices* p'; $Sl \subseteq S$; $p'' \in Sl]\!] \Longrightarrow$ *reachable-through* p S p''

definition *truncation* **where** *truncation* p $S \equiv \{p'$. *reachable-through* p S $p'\}$

lemma *l13*:
  assumes *quorum-of* p Q **and** $p \in W$ **and** *reachable-through* p Q p'
  shows *quorum-of* p' Q
  using *assms* **using** *quorum-assm reachable-through.cases*
  by (*metis is-pbqs subset-iff*)

lemma *l14*:
  assumes *quorum-of* p Q **and** $p \in W$
  shows *quorum* (*truncation* p Q)
proof −
  have $\exists$ $S \in$ *slices* p' . $\forall$ $q \in S$. *reachable-through* p Q q **if** *reachable-through* p Q p' **and** $p' \in W$ **for** p'
    by (*meson assms l13 quorum-is-quorum-of-some-slice stellar.reachable-through.intros*(*2*) *stellar-axioms that*)
  thus *?thesis*
    by (*metis IntE mem-Collect-eq stellar.quorum-def stellar-axioms subsetI truncation-def*)
qed

lemma *l15*:
  assumes *is-cons-cluster* I **and** *quorum-of* p Q **and** *quorum-of* p' Q' **and** $p \in I$ **and** $p' \in I$ **and** $Q \cap Q' \cap W \neq \{\}$
  shows $W \cap$ (*truncation* p Q) $\cap$ (*truncation* p' Q') $\neq \{\}$
proof −
  have *quorum* (*truncation* p Q) **and** *quorum* (*truncation* p' Q') **using** *l14 assms is-cons-cluster-def* **by** *auto*
  moreover have *quorum-of-set* I (*truncation* p Q) **and** *quorum-of-set* I (*truncation* p' Q')

using *mem-Collect-eq quorum-def quorum-of-def quorum-of-set-def reachable-through.intros*(*1*) *truncation-def*
by (*metis Int-iff assms*(*4*) *calculation*(*1*), *metis Int-iff assms*(*5*) *calculation*(*2*))
moreover note ‹*is-cons-cluster I*›
ultimately show *?thesis* **unfolding** *is-cons-cluster-def* **by** *auto*
qed

end

2.3 Elementary quorums

In this section we define the notion of elementary quorum, which is a quorum that has no strict subset that is a quorum. It follows directly from the definition that every finite quorum contains an elementary quorum. Moreover, we show that if Q is an elementary quorum and n_1 and n_2 are members of Q, then n_2 is reachable from n_1 in the directed graph over participants defined as the set of edges (n, m) such that m is a member of a slice of n. This lemma is used in the companion paper to show that probabilistic leader-election is feasible.

locale *elementary* = *stellar*
begin

definition *elementary* **where**
elementary s ≡ *quorum s* ∧ (∀ *s'* . *s'* ⊂ *s* ⟶ ¬*quorum s'*)

lemma *finite-subset-wf*:
shows *wf* {(*X*, *Y*). *X* ⊂ *Y* ∧ *finite Y*}
by (*metis finite-psubset-def wf-finite-psubset*)

lemma *quorum-contains-elementary*:
assumes *finite s* **and** ¬ *elementary s* **and** *quorum s*
shows ∃ *s'* . *s'* ⊂ *s* ∧ *elementary s'* **using** *assms*
proof (*induct s rule*:*wf-induct*[**where** *?r*={(*X*, *Y*). *X* ⊂ *Y* ∧ *finite Y*}])
case *1*
then show *?case* **using** *finite-subset-wf* **by** *simp*
next
case (*2 x*)
then show *?case*
by (*metis* (*full-types*) *elementary-def finite-psubset-def finite-subset in-finite-psubset less-le psubset-trans*)
qed

inductive *path* **where**
path []
| ⋀ *x* . *path* [*x*]
| ⋀ *l n* . ⟦*path l*; *S* ∈ *Q* (*hd l*); *n* ∈ *S*⟧ ⟹ *path* (*n*#*l*)

theorem *elementary-connected*:

```
  assumes elementary s and n₁ ∈ s and n₂ ∈ s and n₁ ∈ W and n₂ ∈ W
  shows ∃ l . hd (rev l) = n₁ ∧ hd l = n₂ ∧ path l (is ?P)
proof –
  { assume ¬?P
    define x where x ≡ {n ∈ s . ∃ l . l ≠ [] ∧ hd (rev l) = n₁ ∧ hd l = n ∧ path
l}
    have n₂ ∉ x using ‹¬?P› x-def by auto
    have n₁ ∈ x unfolding x-def using assms(2) path.intros(2) by force
    have quorum x
    proof –
      { fix n
        assume n ∈ x
        have ∃ S . S ∈ slices n ∧ S ⊆ x
        proof –
           obtain S where S ∈ slices n and S ⊆ s using ‹elementary s› ‹n ∈ x›
unfolding x-def
            by (force simp add:elementary-def quorum-def)
          have S ⊆ x
          proof –
            { assume ¬ S ⊆ x
              obtain m where m ∈ S and m ∉ x using ‹¬ S ⊆ x› by auto
             obtain l′ where hd (rev l′) = n₁ and hd l′ = n and path l′ and l′ ≠ []
                using ‹n ∈ x› x-def by blast
              have path (m # l′) using ‹path l′› ‹m∈ S› ‹S ∈ slices n› ‹hd l′ = n›
                using path.intros(3) by fastforce
               moreover have hd (rev (m # l′)) = n₁ using ‹hd (rev l′) = n₁› ‹l′
≠ []› by auto
              ultimately have m ∈ x using ‹m ∈ S› ‹S ⊆ s› x-def by auto
              hence False using ‹m ∉ x› by blast }
            thus ?thesis by blast
          qed
          thus ?thesis
            using ‹S ∈ slices n› by blast
        qed }
      thus ?thesis by (meson Int-iff quorum-def)
    qed
    moreover have x ⊂ s
      using ‹n₂ ∉ x› assms(3) x-def by blast
    ultimately have False using ‹elementary s›
      using elementary-def by auto
  }
  thus ?P by blast
qed


end
```

2.4 The intact sets of the Stellar Whitepaper

definition *project* **where**
project slices S n $\equiv$ $\{Sl \cap S \mid Sl\ .\ Sl \in slices\ n\}$
— Projecting on S is the same as deleting the complement of S, where deleting is understood as in the Stellar Whitepaper.

2.4.1 Intact and the Cascade Theorem

locale *intact* = — Here we fix an intact set I and prove the cascade theorem.
orig:*stellar slices W*
+ *proj*:*stellar project slices I W* — We consider the projection of the system on I.
for *slices W I* + — An intact set is a set I satisfying the three assumptions below:
assumes *intact-wb*:$I \subseteq W$
and *q-avail*:*orig.quorum I* — I is a quorum in the original system.
and *q-inter*:$\bigwedge$ $Q\ Q'$. $[\![$*proj.quorum Q*; *proj.quorum Q'*; $Q \cap I \neq \{\}$; $Q' \cap I \neq \{\}]\!]$ $\Longrightarrow$ $Q \cap Q' \cap I \neq \{\}$
— Any two sets that intersect I and that are quorums in the projected system intersect in I. Note that requiring that $Q \cap Q' \neq \{\}$ instead of $Q \cap Q' \cap I \neq \{\}$ would be equivalent.
begin

theorem *blocking-safe*: — A set that blocks an intact node contains an intact node. If this were not the case, quorum availability would trivially be violated.
fixes *S n*
assumes $n \in I$ **and** $\forall\ Sl \in slices\ n\ .Sl \cap S \neq \{\}$
shows $S \cap I \neq \{\}$
using *assms q-avail intact-wb* **unfolding** *orig.quorum-def*
by *auto* (*metis inf.absorb-iff2 inf-assoc inf-bot-right inf-sup-aci*(*1*))

theorem *cascade*:
— If U is a quorum of an intact node and S is a super-set of U, then either S includes all intact nodes or there is an intact node outside of S which is blocked by the intact members of S. This shows that, in SCP, once the intact members of a quorum accept a statement, a cascading effect occurs and all intact nodes eventually accept it regardless of what befouled and faulty nodes do.
fixes *U S*
assumes *orig.quorum U* **and** $U \cap I \neq \{\}$ **and** $U \subseteq S$
obtains $I \subseteq S \mid \exists\ n \in I - S\ .\ \forall\ Sl \in slices\ n\ .\ Sl \cap S \cap I \neq \{\}$
proof −
have *False* **if** *1*:$\forall\ n \in I - S\ .\ \exists\ Sl \in slices\ n\ .\ Sl \cap S \cap I = \{\}$ **and** *2*:$\neg(I \subseteq S)$
proof −

First we show that $I - S$ is a quorum in the projected system. This is immediate from the definition of quorum and assumption 1.

have *proj.quorum* $(I-S)$ **using** *1*
by (*simp add*: *proj.quorum-def project-def*) (*metis DiffI IntE IntI empty-iff subsetI*)

Then we show that U is also a quorum in the projected system:

moreover have *proj.quorum U* **using** ‹*orig.quorum U*›
unfolding *proj.quorum-def orig.quorum-def project-def*
by (*simp*; *meson Int-commute inf.coboundedI2*)

Since quorums of I must intersect, we get a contradiction:

ultimately show *False* **using** ‹$U \subseteq S$› ‹$U \cap I \neq \{\}$› ‹$\neg(I \subseteq S)$› *q-inter* **by** *auto*
qed
thus *?thesis* **using** *that* **by** *blast*
qed

end

2.4.2 The Union Theorem

Here we prove that the union of two intact sets that intersect is intact. This implies that maximal intact sets are disjoint.

locale *intersecting-intact* =
i1:*intact slices W* I_1 + *i2*:*intact slices W* I_2 — We fix two intersecting intact sets I_1 and I_2.
+ *proj*:*stellar project slices* $(I_1 \cup I_2)$ *W* — We consider the projection of the system on $I_1 \cup I_2$.
for *slices W* I_1 I_2 +
assumes *inter*:$I_1 \cap I_2 \neq \{\}$
begin

theorem *union-quorum*: *i1.orig.quorum* $(I_1 \cup I_2)$ — $I_1 \cup I_2$ is a quorum in the original system.
using *i1.intact-axioms i2.intact-axioms*
unfolding *intact-def stellar-def intact-axioms-def i1.orig.quorum-def*
by (*metis Int-iff Un-iff le-supI1 le-supI2*)

theorem *union-quorum-intersection*:
assumes *proj.quorum* Q_1 **and** *proj.quorum* Q_2 **and** $Q_1 \cap (I_1 \cup I_2) \neq \{\}$ **and** $Q_2 \cap (I_1 \cup I_2) \neq \{\}$
shows $Q_1 \cap Q_2 \cap (I_1 \cup I_2) \neq \{\}$
— Any two sets that intersect $I_1 \cup I_2$ and that are quorums in the system projected on $I_1 \cup I_2$ intersect in $I_1 \cup I_2$.
proof −

First we show that Q_1 and Q_2 are quorums in the projections on I_1 and I_2.

have *i1.proj.quorum* Q_1 **using** ‹*proj.quorum* Q_1›
unfolding *i1.proj.quorum-def proj.quorum-def project-def*
by *auto* (*metis Int-Un-distrib Int-iff Un-subset-iff*)
moreover have *i2.proj.quorum* Q_2 **using** ‹*proj.quorum* Q_2›
unfolding *i2.proj.quorum-def proj.quorum-def project-def*
by *auto* (*metis Int-Un-distrib Int-iff Un-subset-iff*)

moreover have *i2.proj.quorum* Q_1 **using** ‹*proj.quorum* Q_1›
unfolding *proj.quorum-def i2.proj.quorum-def project-def*
by *auto* (*metis Int-Un-distrib Int-iff Un-subset-iff*)
moreover have *i1.proj.quorum* Q_2 **using** ‹*proj.quorum* Q_2›
unfolding *proj.quorum-def i1.proj.quorum-def project-def*
by *auto* (*metis Int-Un-distrib Int-iff Un-subset-iff*)

Next we show that Q_1 and Q_2 intersect if they are quorums of, respectively, I_1 and I_2. This is the only interesting part of the proof.

moreover have $Q_1 \cap Q_2 \cap (I_1 \cup I_2) \neq \{\}$
if *i1.proj.quorum* Q_1 **and** *i2.proj.quorum* Q_2 **and** *i2.proj.quorum* Q_1
and $Q_1 \cap I_1 \neq \{\}$ **and** $Q_2 \cap I_2 \neq \{\}$
for Q_1 Q_2
proof −
have *i1.proj.quorum* I_2
proof −
have *i1.orig.quorum* I_2 **by** (*simp add*: *i2.q-avail*)
thus *?thesis* **unfolding** *i1.orig.quorum-def i1.proj.quorum-def project-def*
by *auto* (*meson Int-commute Int-iff inf-le2 subset-trans*)
qed
moreover note ‹*i1.proj.quorum* Q_1›
ultimately have $Q_1 \cap I_2 \neq \{\}$ **using** *i1.q-inter inter* ‹$Q_1 \cap I_1 \neq \{\}$› **by** *blast*

moreover note ‹*i2.proj.quorum* Q_2›
moreover note ‹*i2.proj.quorum* Q_1›
ultimately have $Q_1 \cap Q_2 \cap I_2 \neq \{\}$ **using** *i2.q-inter* ‹$Q_2 \cap I_2 \neq \{\}$› **by** *blast*
thus *?thesis* **by** (*simp add*: *inf-sup-distrib1*)
qed

Next we show that Q_1 and Q_2 intersect if they are quorums of the same intact set. This is obvious.

moreover
have $Q_1 \cap Q_2 \cap (I_1 \cup I_2) \neq \{\}$
if *i1.proj.quorum* Q_1 **and** *i1.proj.quorum* Q_2 **and** $Q_1 \cap I_1 \neq \{\}$ **and** $Q_2 \cap I_1 \neq \{\}$
for Q_1 Q_2
by (*simp add*: *Int-Un-distrib i1.q-inter that*)
moreover
have $Q_1 \cap Q_2 \cap (I_1 \cup I_2) \neq \{\}$
if *i2.proj.quorum* Q_1 **and** *i2.proj.quorum* Q_2 **and** $Q_1 \cap I_2 \neq \{\}$ **and** $Q_2 \cap I_2 \neq \{\}$
for Q_1 Q_2
by (*simp add*: *Int-Un-distrib i2.q-inter that*)

Finally we have covered all the cases and get the final result:

ultimately
show *?thesis*

by (*smt* (*verit*, *best*) *Int-Un-distrib Int-commute assms*(*3*) *assms*(*4*) *sup-eq-bot-iff*)
qed

end

end

References

[1] G. Losa, E. Gafni, and D. Mazières. Stellar Consensus by Instantiation. In J. Suomela, editor, *33rd International Symposium on Distributed Computing (DISC 2019)*, volume 146 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.