Cellular Mutual Credit for Survival Exchange Under Institutional Failure

A Bitcoin-Style Whitepaper Draft

Part 1: Abstract, Introduction, Model, and Design Goals

Abstract

We present a protocol for exchanging goods and services without reliance on fiat currency, banks, or centralized issuers. The system operates as a collection of local “cells,” each maintaining a zero-sum mutual-credit ledger with hard debt limits. Participants earn credit by providing goods or services and spend credit by receiving goods or services, with the cell-wide balance sum constrained to zero at all times. The protocol is designed for collapse conditions where formal enforcement is weak, institutions fail, and coordination for survival becomes primary. We formalize the ledger as a constrained state machine, bound one-shot defection gains via hard credit ceilings, and give sufficient conditions for cooperation as a stable equilibrium in a repeated-game setting. We further define an optional federation mechanism allowing capped inter-cell netting while preventing systemic contagion through severability rules. The protocol explicitly avoids features that invite capture or suppression by states or large actors: there is no central issuer, no yield, no global money supply, and no global clearinghouse.

---

1. Introduction

1.1 Motivation

When fiat systems remain functional, most people default to money for exchange because it minimizes transaction friction and because legal enforcement backstops contracts. Under economic collapse—hyperinflation, capital controls, bank failures, widespread unemployment, or state capacity degradation—money can fail as a reliable medium of exchange and as a coordination tool. Yet barter is typically inefficient due to the double coincidence of wants. Historically, communities under stress often reconstitute credit before they reconstitute cash.

This paper proposes a mutual-credit protocol intended to support survival-level exchange in adverse conditions. The design target is not “a better currency.” The target is a local-first coordination substrate that:

- permits multi-party exchange without fiat,
- limits extractive behavior by bounding default payoffs,
- remains functional under partial connectivity and censorship pressure,
- scales by replication and severability rather than by global aggregation.

1.2 Why “Cellular”

A global ledger is a single point of failure: technically, economically, and politically. If one network becomes the exchange substrate for a large population, it becomes:

- a censorship target,
- a capture target,
- a systemic-risk amplifier.

We therefore make cells the primary unit: small, geographically compact groups with high interaction frequency and credible local enforcement. Cells can trade internally with high trust density; optional federation allows limited inter-cell exchange without allowing one cell to export unlimited risk to others.

1.3 Design Priorities

We optimize for:

1. Survivability: ability for participants to obtain essentials via local exchange.
2. Incentive stability: defection is bounded and dominated under repeated interaction.
3. Severability: failures are local; contagion is prevented by caps and isolation.
4. Low capture surface: no global issuer, no yield, no speculative token, no “bank-like” behavior.
5. Local legitimacy: dispute resolution and enforcement live at the edge, not in opaque algorithms.

---

2. System Model and Assumptions

2.1 Participants and Cells

A cell C is a set of N participants:

C = \{1,2,\dots,N\}

Cells are intended to be geographically compact and socially legible (not necessarily “friends,” but within a realistic trust-and-reputation radius).

2.2 Ledger Balances

Each participant i has a balance b_i(t)\in\mathbb{R} at discrete time t. Positive balances represent net provision to the cell; negative balances represent net receipt from the cell.

Zero-sum constraint (cell conservation law):

\sum_{i\in C} b_i(t) = 0 \quad \forall t

This is the defining feature of mutual credit: the “money” is not an asset issued by a central entity; it is a ledger of net obligations between participants.

2.3 Unit of Account

We adopt a unit of account designed to reduce speculation and to remain meaningful under collapse. Default choice for this draft:

- 1 unit = 1 hour of median local labor (a “labor-hour” numeraire).

This is not claimed to be perfectly stable, but it is:

- locally interpretable,
- difficult to financialize,
- aligned with survival exchange (work and goods tied to labor constraints).

(Alternative numeraires—calorie equivalents or a basket—will be treated later as variants.)

2.4 Transactions

A transaction transfers value $v > 0$ from buyer p to seller q, updating balances:

$$b_p \leftarrow b_p - v,\quad b_q \leftarrow b_q + v$$

This preserves the zero-sum constraint automatically.

2.5 Hard Debt Limits

Every participant i has a hard debt limit $L_i(t) > 0$, and the system enforces:

$$b_i(t) \ge -L_i(t)$$

This bound is central. It caps the maximum net extraction a participant can take before losing access, thereby bounding defection payoffs.

We allow $L_i(t)$ to vary only within hard global bounds:

$$L_{\min} \le L_i(t) \le L_{\max}$$

and only via transparent local rules (e.g., reputation tiers or council decisions). The protocol must never permit unlimited negative balances.

2.6 Commitments and Future-Dated Services

Survival economies rely on scheduled obligations (food rotations, childcare shifts, repair duties). A protocol that only supports instantaneous spot trades will fail under real conditions.

We model commitments as obligations o due at time τ:

$$o = (p \rightarrow q,\ v,\ \tau)$$

Two modes exist:

- Soft commitments: obligations are recorded but not escrowed; enforcement is social/governance-only.
- Escrowed commitments: the system reserves credit capacity for future delivery, preventing over-extension.

We will specify both. The core ledger remains deterministic either way; commitments are auxiliary.

2.7 Threat Model (Preview)

We assume adversarial behavior is present:

- Opportunistic defection: default and exit.
- Partial shirking: degrade quality or delay.
- Collusion: reputation rings, coordinated looting.
- Sybil attempts: multiple identities to bypass limits.
- External interference: censorship, legal pressure, infiltration by large actors.

We do not assume courts are available or effective. The protocol therefore must be robust under weak enforcement, relying primarily on bounded loss and severability.

---

3. Design Goals as Formal Properties

We now convert the motivations into explicit properties we want the protocol to satisfy.

Property P1: Conservation (Cell Zero-Sum)

For all admissible state transitions,

\sum_{i\in C} b_i(t) = 0

This prevents hidden issuance, political capture via printing, and “credit inflation” inside a cell.

Property P2: Bounded One-Shot Extraction

For any participant i, the maximum net value extractable before exclusion is bounded above by a function of L_i. Informally:

\text{DefectionGain}_i \lesssim L_i

This creates a tunable “security budget” through L_i, analogous in spirit to how Bitcoin bounds adversarial advantage via economic cost, but here the bound is direct.

Property P3: Cooperation Compatibility Under Repetition

There exist parameter regimes (N, L, \text{enforcement}) such that cooperation is a stable equilibrium in a repeated game with imperfect information. Concretely, we require sufficient conditions under which:

- cooperating yields higher discounted utility than defecting,
- defection triggers exclusion with bounded loot,
- the system remains trade-liquid after shocks.

Property P4: Contagion Resistance via Severability

Failure of a participant should not collapse the cell; failure of a cell should not collapse the federation. This requires:

- bounded individual default impact,
- bounded inter-cell exposure,
- automatic isolation when constraints violated.

Property P5: Low Capture Surface

The protocol avoids structures that invite suppression or capture:

- no central issuer,
- no yield or interest,
- no global settlement layer required for survival,
- no single administrative entity required to operate.

These are not just political choices; they reduce game-theoretic attack surfaces by removing privileged rent-extraction positions.

---

4. Roadmap of the Paper

Next parts will:

- specify the protocol precisely (state machine + admissible transitions),
- prove conservation and bounded extraction,
- derive equilibrium conditions for cooperation,
- define federation caps and severability rules,
- give parameter selection guidance,
- define a simulation-based validation suite.

Part 2: Protocol Specification

Ledger rules, transaction validity, commitments, identity/admission, governance hooks

---

5. Protocol Overview

The protocol consists of independent cells C. Each cell maintains:

1. A mutual-credit ledger with hard constraints.
2. A membership set (identity/admission).

3. A local rulebook that may tune parameters within global bounds.
4. Optional commitment records for future-dated obligations.
5. Optional federation links to other cells for capped inter-cell exchange.

The core protocol is intentionally minimal: it defines invariants and admissible transitions that any implementation must satisfy.

---

6. Cell Ledger: State and Invariants

6.1 State Variables

For a cell C at time t, define the state:

- Membership:

  M(t) \subseteq \mathcal{I}

  where \mathcal{I} is the space of admissible identities and |M(t)|=N.

- Balances:

  b(t) = \{b_i(t)\}_{i\in M(t)},\quad b_i(t)\in \mathbb{R}

- Debt limits:

  L(t) = \{L_i(t)\}_{i\in M(t)},\quad L_i(t)\in [L_{\min},L_{\max}]

- (Optional) Reserved capacity for commitments:

  r(t)=\{r_i(t)\}_{i\in M(t)},\quad r_i(t)\ge 0

- Reputation signal (auxiliary; not required for conservation):

  R(t)=\{R_i(t)\}_{i\in M(t)},\quad R_i(t)\in [0,1]

- Event log E(t): append-only record of validated transactions and commitments.

6.2 Core Invariants (Must Hold Always)

I1. Conservation (zero-sum):

\sum_{i\in M(t)} b_i(t) = 0

I2. Hard debt floor:

b_i(t)\ge -L_i(t)\quad \forall i

I3. Commitment safety (if escrowed commitments are enabled):

b_i(t)-r_i(t)\ge -L_i(t)\quad \forall i

I4. No negative reserves:

r_i(t)\ge 0

These invariants are enforced by transaction validation rules (next section).

---

7. Transaction Semantics

7.1 Transaction Type T1: Spot Exchange

A spot exchange is a tuple:

x = (p,q,v,\text{meta})

where:

- p = payer/buyer (receives goods/services)
- q = payee/seller (provides goods/services)
- v>0 = value in cell units
- meta includes timestamps, optional proofs, and human-readable description.

Validation Rules for T1

A spot transaction is valid if:

V1. Membership:

$p,q \in M(t)$

V2. Positive value:

$v>0$

V3. Debt-limit feasibility:

- If commitments are not escrowed:

  $b_p(t)-v \ge -L_p(t)$

- If escrowed commitments are enabled:

  $b_p(t)-r_p(t)-v \ge -L_p(t)$

State Update

If valid, balances update as:

$$b_p(t+1)=b_p(t)-v,\quad b_q(t+1)=b_q(t)+v$$

All other b_i unchanged. Reserves r_i unchanged.

Lemma (Conservation): The update preserves invariant I1 automatically.

---

8. Commitments (Survival Scheduling Primitive)

Commitments allow the system to support planned production/distribution (food rotas, maintenance shifts), which is necessary for “live off the app” operation.

8.1 Commitment Type C0: Soft Commitment (record only)

A soft commitment is:

$$o=(p\rightarrow q,\ v,\ \tau,\ \text{meta})$$

It records intent but does not constrain spending capacity.

- Creation rule: any p,q\in M(t), v>0, \tau in the future.
- Fulfillment is executed later by a T1 spot exchange after delivery.

Pros: simplest, low friction.

Cons: higher default risk.

8.2 Commitment Type C1: Escrowed Commitment (reserve capacity)

Escrowed commitment adds reserve mechanics:

- On creation at time t, reserve increases:

  r_p(t+1)=r_p(t)+v

- Must satisfy feasibility at creation:

  b_p(t)-r_p(t)-v \ge -L_p(t)

- On fulfillment at time \tau, execute the transaction and release reserve:

  b_p \leftarrow b_p - v,\quad b_q \leftarrow b_q + v,\quad r_p \leftarrow r_p - v

Pros: prevents over-commitment; reduces “exit scam” capacity.

Cons: reduces flexibility; can feel restrictive.

8.3 Cancellation / Dispute Hooks

Commitments can be cancelled only under:

- mutual consent, or
- governance decision (see Section 10)

Cancellation updates:

- Soft: remove record.
- Escrowed: release reserve r_p \leftarrow r_p - v.

---

9. Identity, Membership, and Sybil Resistance (Cell-Level)

A collapse-resilient system cannot assume global identity services. We therefore specify membership as local admission rather than global KYC. The goal is not perfect identity; it is bounded damage and credible local exclusion.

9.1 Identity Primitive

Each participant has a cell-scoped identity ID_i represented by a public key (or equivalent):

- signing key for transactions
- optional metadata (name, location, skills)

Importantly:

- identities are cell-scoped by default
- cross-cell identity linking is optional, not required

9.2 Admission Rule (Minimal)

A cell admits a new identity if:

- governance quorum approves (Section 10), and
- the cell assigns initial parameters (b_i=0, L_i=L_{\text{init}}, R_i=R_{\text{init}}).

9.3 Deposit / Stake (Optional, Non-Fiat)

To reduce Sybil attacks, cells may require an admission “bond” in one of these forms:

- service bond: newcomer must complete k hours of tasks before receiving full L
- sponsor bond: an existing member sponsors and shares downside via reduced L if the newcomer defects
- physical bond: locally held goods/tool deposit (only feasible in tight communities)

The protocol supports these as policy, not as a global requirement.

9.4 Exclusion Rule

A participant is automatically restricted if:

b_i(t) = -L_i(t)

At the hard floor, they cannot initiate further payments. The cell may also:

- freeze them (governance),
- require restitution plans (governance),
- expel them (governance).

This is the primary enforcement lever in weak-rule-of-law environments: access control.

---

10. Governance Hooks (Local, Bounded Authority)

The protocol does not dictate politics; it dictates constraint-respecting powers.

10.1 Governance Object

Each cell maintains a governance process with:

- a council set G(t)\subseteq M(t) (e.g., 5–9 members),

- a quorum rule q (e.g., majority or supermajority),
- a term/rotation policy.

10.2 Allowed Governance Actions (Must Respect Invariants)

Governance may:

1. Admit/expel members.
2. Freeze accounts pending dispute resolution.
3. Adjust L_i(t) within [L_{\min},L_{\max}].
4. Adjust reputation R_i(t) by transparent procedures.
5. Authorize commitment cancellations.
6. Define local policies (task categories, scheduling norms).

Governance may not:

- create net credit (cannot violate conservation),
- override the hard floor,
- seize balances to create net positive issuance,
- promise convertibility or yield.

Formally, any governance action must preserve I1–I4.

10.3 Dispute Resolution Primitive

A dispute is an object:

d=(i,j,\text{claim},\text{evidence},\text{status})

Outcomes can include:

- require a compensating transaction,
- freeze one or both parties,
- adjust L and/or R,
- expulsion.

The protocol's job is to make disputes survivable: even if adjudication is imperfect, bounded extraction prevents catastrophic loss.

---

11. Implementation Notes (Protocol-Level, Not UI)

11.1 Local-First Operation

A cell must be able to function:

- with intermittent internet,
- without external payment rails,
- without a central server.

This implies implementations should support:

- local storage,
- eventual consistency within the cell,
- cryptographic signing of events.

11.2 Deterministic Core

The validation rules for T1/C0/C1 are deterministic and auditable. Any “AI layer” must be read-only advisory and must not modify balances or limits outside governance.

---

12. What Comes Next

Part 3 will formalize:

- the defection bound G_i \le b_i+L_i,
- sufficient conditions for cooperation equilibrium in repeated interaction,
- why cell size and hard limits matter mathematically,
- and the beginnings of federation (inter-cell netting constraints).

Part 3: Incentives and Equilibrium Analysis

Core math: bounded defection, cooperation conditions, and stability metrics

---

13. Economic Security as Bounded Defection

The protocol’s primary security primitive is the hard debt floor:

b_i(t)\ge -L_i(t)

This creates a hard upper bound on how much any participant can extract from the cell without reciprocating.

13.1 Proposition 1 (Bounded One-Shot Extraction)

Assume:

- spot exchanges (T1) are the only mechanism that changes balances, and

- a participant i who defects will be excluded once they reach the hard floor b_i=-L_i (or earlier).

Let b_i(t_0) be i's balance at the onset of defection. Let G_i be the maximum net value i can receive from others (net consumption) before exclusion.

Then:

G_i \le b_i(t_0)+L_i(t_0)

Proof sketch. Each unit of value received decreases b_i by that amount (buyer pays by becoming more negative). The most negative b_i can become is -L_i. Thus total additional decrease in b_i is at most b_i(t_0)-(-L_i)=b_i(t_0)+L_i. ∎

Corollary (Worst-case looting bound). If participants typically start near b_i\approx 0, then:

G_i \lesssim L_i

So L is the tunable "loss cap" per identity.

13.2 Why This Matters

In collapse, formal enforcement is weak. You cannot rely on courts to punish default. Therefore the system must be designed so that the maximum damage any rational defector can do is bounded and small enough that cooperation remains rational.

---

14. The Repeated Game: When Cooperation Is Rational

We now model interaction as repeated trade within a cell.

14.1 Setup

Each period, an agent derives utility from access to goods/services mediated by the cell. Let:

- u_i^C = per-period expected utility for agent i when cooperating (continued membership)
- u_i^D = one-time utility from defection (bounded by G_i, itself bounded by L_i)
- \delta_i \in (0,1) = discount factor capturing continuation probability, patience, and expected survival horizon within the cell

The discounted present value of cooperation is approximately:

$$V_i^C = \sum_{t=0}^{\infty} \delta_i^t u_i^C = \frac{u_i^C}{1-\delta_i}$$

Defection gives:

$$V_i^D \approx u_i^D$$

(plus possibly post-exclusion outside options; we incorporate those below.)

14.2 Proposition 2 (Sufficient Cooperation Condition)

Cooperation is individually rational (a best response) if:

$$\frac{u_i^C}{1-\delta_i} \ge u_i^D$$

Using the defection bound $u_i^D \le \alpha (b_i+L_i)$ for some conversion factor α from ledger units to utility, a sufficient design condition is:

$$L_i \le \frac{u_i^C}{\alpha(1-\delta_i)} - b_i$$

If we take $b_i\approx 0$ and normalize $\alpha=1$ in "utility-equivalent units," a clean design inequality emerges:

$$\boxed{L \le \frac{u^C}{1-\delta}}$$

Interpretation.

- Larger u^C (cell provides essential value) → can tolerate larger L.
- Smaller $(1-\delta)$ (agents expect to keep interacting) → can tolerate larger L.
    - If collapse reduces δ (people plan to flee) → you must reduce L.

14.3 Outside Options and "Exit Attacks"

Let w_i be outside-option utility after exclusion (barter, theft, state rations, etc.). Then:

$$V_i^D \approx u_i^D + w_i$$

Cooperation condition becomes:

$$\frac{u_i^C}{1-\delta_i} \ge u_i^D + w_i$$

This is crucial: if outside options are good (e.g., a functioning parallel market), defection is more attractive, so L must be tighter or membership benefits must be higher.

Collapse paradox: severe collapse often reduces outside options w_i, which makes cooperation easier—if the cell can supply essentials.

---

15. Liquidity and Seller Acceptance: Not Just “Theft”

Mutual credit can fail without explicit defection if participants refuse to accept credits due to fear they won’t be able to spend them later. This is a liquidity/acceptance problem, which we can model via balance dispersion.

15.1 Balance Dispersion Metrics

Let the mean balance be \bar b = 0 by conservation. Define variance:

\sigma^2(t) = \frac{1}{N}\sum_{i\in C} b_i(t)^2

Also define debt concentration:

D(t)=\max_{i\in C}(-b_i(t))

and credit concentration:

P(t)=\max_{i\in C} b_i(t)

High D(t) indicates one or more participants have accumulated large debt near their limit; sellers may fear those debts will never be worked off.

15.2 A Minimal Acceptance Model

Suppose sellers accept credits from buyer p with probability A_p that decreases as p approaches their debt limit:

A_p = f\!\left(\frac{b_p}{L_p}\right),\quad f \text{ decreasing in } -\frac{b_p}{L_p}

A simple linear form:

A_p = 1 - \gamma\left(\frac{-b_p}{L_p}\right),\quad \gamma\in(0,1]

When b_p=-L_p, acceptance collapses.

This implies the system’s trade throughput collapses not only when defection happens, but when many participants drift toward the hard floor.

15.3 Design Implication: You Need “Debt Work-Off” Dynamics

Survival cells require mechanisms (not necessarily coercive) to pull participants away from the hard floor:

- visibility of high-need tasks,
- priority matching for debtors to earn credits,
- temporary throttles on further spending (already enforced),
- and governance nudges (task rotation, obligations).

This is why the system is fundamentally a coordination/scheduling layer as much as a ledger.

---

16. Collusion and Sybil: Why Cell Size Matters

16.1 Sybil Impact Bound (If Admission Is Controlled)

If a cell requires admission (Section 9), the attacker’s effective number of identities S is bounded by admission friction. The maximum total extraction is then:

G_{\text{attack}} \le \sum_{s=1}^{S} (b_s + L_s) \approx S L

Thus, the key is to keep S small. This is achieved by:

- local admission,
- sponsor bonds,
- service bonds.

16.2 Collusive Reputation Rings

If reputation is used to raise L_i, colluders can try to inflate one another’s reputations to increase limits and then loot.

Therefore, any rule that increases L_i based on reputation must satisfy:

$$L_i(t) \le L_{\max} \quad \text{and} \quad \Delta L_i \text{ is slow}$$

and increases must be rate-limited:

$$L_i(t+1)-L_i(t) \le \eta$$

for small η. This prevents rapid “pump then dump” attacks.

---

17. A Practical Parameter Inequality Set

We now have three core inequalities guiding safe parameter choices:

(A) Individual extraction bound

$$G_i \lesssim L$$

(B) Cooperation condition

$$L \le \frac{u^C}{1-\delta} - w$$

(where w is outside option in equivalent units)

(C) Sybil-limited extraction

$$G_{\text{attack}} \lesssim S L$$

so you must enforce small S via admission friction.

These define the “security envelope” of the system.

---

18. What Comes Next

Part 4 will define:

- cell optimality more sharply (trade matching vs governance cost),
- federation math: inter-cell netting, exposure caps, and severability proofs,

- and the explicit anti-contagion mechanism that makes the system resilient at scale-by-replication.

Part 4: Federation, Roll-Up Constraints, and Severability

Inter-cell exchange without systemic leverage or contagion

---

19. Why Federation Exists (and Why It Must Be Capped)

A purely local cell can be robust but may lack:

- specialization (medical, fabrication),
- seasonal goods,
- redundancy against local shocks.

Federation enables limited trade across cells. However, federation is the primary route to systemic failure: if cells can run persistent net deficits with others, the network recreates the very leverage dynamics it sought to escape.

Therefore the federation layer must satisfy two properties:

- F1: Exposure bounds — no cell can accumulate large net obligations to the rest of the network.
- F2: Severability — any cell can be isolated without breaking internal accounting of other cells.

---

20. Federation Model

Let cells be indexed $k \in \{1,\dots,m\}$.

Cell k has membership M_k, balances $\{b_i\}_{i\in M_k}$, and debt limits $\{L_i\}$.

20.1 Two-Ledger Decomposition

We keep the core idea: local ledgers are primary. Federation is not a global ledger; it is a limited netting layer.

Define for each cell k an external position $B_k(t)$, representing the net claim of cell k on other cells (positive) or net obligation (negative). We require:

$$\sum_{k=1}^m B_k(t) = 0 \quad \forall t$$

This is the federation analogue of conservation.

Key design choice: $B_k(t)$ is not a freely spendable "cell currency." It is a bookkeeping constraint determining how much cross-cell trade is allowed.

20.2 Inter-Cell Transaction

Suppose a member $p \in M_a$ in cell a buys a good/service from member $q \in M_b$ in cell b with value v.

We implement this as two simultaneous local transactions plus a federation update:

1. In cell a: p pays v to a local "external clearing account" X_a.

   $$b_p \leftarrow b_p - v,\quad b_{X_a} \leftarrow b_{X_a} + v$$

2. In cell b: a local "external clearing account" X_b pays v to q.

   $$b_{X_b} \leftarrow b_{X_b} - v,\quad b_q \leftarrow b_q + v$$

3. Federation positions update:

   $$B_a \leftarrow B_a - v,\quad B_b \leftarrow B_b + v$$

Interpretation:

- Cell a has imported v from outside and owes the federation v.
- Cell b has exported v and is owed v.

20.3 Conservation Holds Locally and Federally

- Each cell remains zero-sum internally because X_a and X_b are members of their respective ledgers.
- Federation remains zero-sum because B_a and B_b update oppositely.

Thus federation does not break the mutual credit principle; it composes it.

---

21. Exposure Caps (The Anti-Leverage Constraint)

21.1 Cell Capacity Scale

Within cell k, define aggregate credit capacity:

\Lambda_k = \sum_{i\in M_k} L_i

If all L_i=L, then \Lambda_k = N_k L.

This quantity measures the maximum total net extraction potential inside the cell. Federation exposure should be some small fraction of this; otherwise the cell becomes systemically indebted externally.

21.2 Federation Cap Rule

Define an exposure cap parameter \beta \in (0,1) (small). Enforce:

\boxed{|B_k(t)| \le \beta \Lambda_k}

If all members have equal limits:

|B_k(t)| \le \beta N_k L

Typical safe regime:

\beta \in [0.05, 0.15]

21.3 Why This Works (Intuition)

- If \beta is small, inter-cell deficits cannot become large enough to threaten cell survival.
- A cell that imports too much is forced to export later (or stop importing).
- There is no pathway to create “federation debt empires.”

This is the core mathematical barrier preventing a mutual credit federation from turning into a shadow central bank.

---

22. Severability and Contagion Prevention

22.1 Definition (Severability)

A federation is severable if, for any subset of cells S, the remainder S^c can continue operating internally without needing settlement from S.

We want:

- internal ledgers stay valid,
- limits still enforce bounded extraction,
- federation connections can be cut without breaking conservation within remaining cells.

22.2 Proposition 3 (Severability Under Exposure Caps)

Assume:

1. Each cell ledger satisfies invariants I1–I4.
2. Federation exposure caps |B_k|\le \beta \Lambda_k are enforced.
3. Inter-cell trade is implemented via clearing accounts X_k as in Section 20.

Then if any cell j is isolated (no further federation trades), all other cells remain internally consistent and can continue trading internally without needing any balance updates from j.

Proof sketch.

Isolation stops further updates involving B_j and other B_k. Since each cell's internal conservation is maintained by its own ledger updates (including its clearing account), and no other cell requires settlement from j to preserve internal conservation, the ledgers remain valid. Federation claims against j remain recorded in B, but are bounded for each cell by \beta \Lambda_k, limiting harm and preventing collapse. **I**

22.3 Practical Isolation Rule (Automatic Quarantine)

If a cell violates cap constraints or exhibits high-risk signals:

- federation edge is suspended
- cell can continue internally
- re-connection requires returning within bounds

Formally: if |B_k|>\beta \Lambda_k, set federation adjacency A_{k\ell}=0 for all \ell.

---

23. Federation Liquidity: Preventing "Frozen Claims"

Even with caps, exporters may worry that their positive federation position B_k>0 is not spendable.

We therefore need a spending rule:

23.1 Spendability Constraint

Cell k may import (increase deficit, decrease B_k) up to the cap. Exporters (with B_k>0) can always import later unless:

- they choose not to,
- they are politically/physically blocked,
- or the federation is severed.

This is a real risk under collapse. We mitigate it with:

- multi-edge federation (each cell connects to several others),
- low \beta (claims are small and non-fatal),
- preference for reciprocal trade pairs (reduce net accumulation).

23.2 Anti-Hoarding / Netting Incentive

You may optionally impose a mild decay fee on large positive B_k beyond a threshold to encourage circulation, but this is politically delicate and may resemble “tax.” In the collapse use-case, the simpler solution is:

- keep \beta small,
- encourage reciprocal exchange.

(We can formalize an optimal control here later if you want, but I’d default to simplicity.)

---

24. Cell Size and Federation Degree (Graph Structure)

24.1 Federation Graph

Model federation as a graph G=(V,E) where vertices are cells and edges are allowed trading relationships.

Each cell should have degree d (number of federation partners) large enough for redundancy, small enough for legibility and governance:

d \in [3,7]

This provides:

- alternate routes if one cell is isolated,

- diversified supply,
- limited exposure and monitoring burden.

24.2 Risk Propagation Bound

In a network with exposure caps \beta, worst-case loss to a cell from immediate neighbors is bounded by:

\text{Loss}_k \le d \cdot \beta \Lambda_k

So choosing small d and small \beta bounds contagion even under correlated neighbor failure.

This is the federation analogue of limiting leverage + limiting counterparty concentration in financial risk management.

---

25. Summary: Federation Rules as Mathematical Guardrails

Federation is safe if and only if:

1. Local ledgers remain primary.
2. Cross-cell trades are routed through clearing accounts and a separate B_k position.
3. Exposure caps are enforced:

   |B_k| \le \beta \Lambda_k

4. Isolation is automatic and does not break internal operations.
5. Federation degree d is bounded to limit counterparty concentration.

These make the system scale by replication, not by consolidation.

---

26. What Comes Next

Part 5 will address the remaining big math gap:

- optimal parameterization: choosing N, L, \beta, admission friction, and reputation dynamics as a coherent set,
- plus a formal “survival constraint layer” (task coverage model) so a person can realistically live off the network.

Part 5: Parameterization and Survival-Coverage Mathematics

Choosing N, L, \beta coherently, and modeling “can a person survive on this?”

---

27. Parameterization as a Constrained Design Problem

We want parameters that simultaneously satisfy:

1. Bounded loss (security): L small enough that looting is tolerable
2. Cooperation equilibrium: L large enough that newcomers can function and trade isn’t strangled
3. Liquidity: balances don’t concentrate at -L across many participants
4. Sybil resistance: effective attacker identities S remain small
5. Federation safety: \beta small enough to prevent contagion but large enough for specialization trade
6. Survival feasibility: essential needs can be met by scheduled production + exchange

This is best treated as a feasibility region in parameter space, not a single “optimum.”

---

28. The Two Core Scales: Individual Limit and Cell Capacity

28.1 Individual Credit Limit L

L is the maximum net draw a person can make before they must “work back” or stop spending.

28.2 Aggregate Cell Capacity \Lambda

Define:

\Lambda = \sum_{i\in C} L_i

If equal limits, \Lambda = NL.

\Lambda bounds:

- maximum simultaneous net “claims” by creditors,
- maximum internal exposure to defaults,
- maximum internal trade imbalance.

This matters for both liquidity and federation caps.

---

29. Calibrating L from Survival Economics

To make “survive off the app” operational, we need a baseline for essential consumption.

29.1 Normalize Units to Labor-Hours

We take:

- 1 unit = 1 hour of median local labor

Let a person's essential weekly needs require h_E labor-hours from the cell (food production/distribution, fuel logistics, basic maintenance, childcare, etc.) net of their own contributions.

In a functioning survival cell, a typical adult can contribute h_W hours/week (bounded by health, shock conditions). In collapse, h_W may increase (more time available) but productivity may fall (scarcity).

29.2 Practical Limit Range in This Numeraire

A useful principle:

- L should cover a short survival bridge (to avoid immediate starvation spirals)
- but not long enough to make default profitable

So pick:

\boxed{L \approx \theta \, h_E}

with:

\theta \in [0.5, 2]

Interpretation:

- \theta=0.5: tight discipline; low default loss, higher friction
- \theta=1: one week of essential net support
- \theta=2: two-week bridge; easier onboarding, higher looting bound

Collapse-hardened default: \theta \approx 1.

So:

\boxed{L \approx h_E \text{ (about one week of essential needs)}}

This ties your "security budget" to survival reality, not arbitrary numbers.

---

30. Cooperation Condition Restated in Survival Terms

From Part 3, cooperation is sustained if:

\frac{u^C}{1-\delta} \ge u^D + w

We now approximate these in labor-hour terms.

- u^D \lesssim L (bounded extraction)
- u^C: value of continued membership. If the cell enables survival, u^C is on the order of “weekly essential access” in utility units.
- w: outside options (barter/theft/rations). In severe collapse, w is often low and risky.

If we normalize “one week of essentials” as utility 1, then:

- u^C \approx 1
- u^D \approx \theta (weeks of looting)
- w maybe 0.2–0.5 (outside options are bad)

Then cooperation requires:

\frac{1}{1-\delta} \gtrsim \theta + w

Example: if \delta=0.8, then 1/(1-\delta)=5. That can sustain \theta up to ~4-ish. If \delta=0.6, then it’s only 2.5; you need tighter L and/or stronger local enforcement.

Design insight: In “panic flight” regimes where \delta drops, the system must automatically tighten L and increase admission friction, or the incentive equilibrium flips.

So you want a policy:

- default L set for normal times,
- emergency mode reduces L for new accounts and increases bonding.

We can formalize emergency mode later as a state-dependent control, but the math supports it.

---

31. Cell Size N: Matching Efficiency vs Enforcement Bandwidth

31.1 Matching Efficiency

Let the diversity of skills scale roughly like:

\text{SkillCoverage}(N) \approx 1 - e^{-kN}

for some k>0 (diminishing returns).

31.2 Enforcement Bandwidth / Governance Load

Disputes, monitoring, and social legibility degrade with N. A crude but useful scaling:

- potential pairwise interactions: O(N^2)
- dispute cases often scale with transaction volume, which rises with N

You get a tradeoff: beyond some N, dispute load grows faster than trust and coordination.

31.3 Parameter Recommendation (Now in “why” form)

The feasible region typically falls in:

\boxed{N \in [50,150]}

and the “survival kernel” sweet spot is often:

\boxed{N \in [60,100]}

because:

- enough labor-hours exist for essential roles redundancy,
- enough skill diversity exists,

- yet governance remains human-manageable.

---

32. Federation Cap \beta: Linking External Exposure to Survival Reserves

From Part 4:

|B_k| \le \beta \Lambda_k

Since \Lambda_k \approx N_k L, federation exposure is bounded by a fraction of “total cell credit capacity.”

Interpretation: \beta\Lambda is the maximum amount the cell can be net-importing from the outside (or net-owed).

To keep federation from becoming a dependency:

- \beta must be small enough that if federation collapses, the cell still survives.

Thus choose \beta based on what fraction of essentials can safely be imported.

Let f be the fraction of essential needs met externally in steady-state. In collapse planning, you want f small (local resilience). Then:

\boxed{\beta \approx f}

Typical robust choices:

\boxed{\beta \in [0.05,0.15]}

- 0.05: almost fully local; federation is icing
- 0.15: meaningful specialization trade but still non-systemic

---

33. Survival Coverage as a Constraint Satisfaction Problem

This is the missing piece that makes the app more than a ledger.

33.1 Essential Task Set

Define a set of essential task categories \mathcal{T}, e.g.:

- food production & distribution
- water/fuel logistics
- shelter repair
- sanitation
- basic medical support
- security/coordination
- childcare

Each task type t \in \mathcal{T} requires a minimum labor-hours per period (say weekly) to keep the cell viable:

H_t^{\min} \quad \text{hours/week}

Total essential labor demand:

H^{\min} = \sum_{t\in \mathcal{T}} H_t^{\min}

33.2 Labor Supply

Each participant i has labor supply capacity s_i (hours/week) that they can contribute, and skill suitability weights a_{i,t} \in [0,1] indicating effectiveness at task t.

We choose allocations x_{i,t} \ge 0 representing hours assigned by person i to task t, subject to:

\sum_{t} x_{i,t} \le s_i \quad \forall i

and coverage constraints:

$$\sum_i a_{i,t} x_{i,t} \ge H_t^{\min} \quad \forall t$$

This is a feasibility problem. The cell is survivable if there exists x satisfying these.

33.3 Why This Integrates with the Ledger

The ledger becomes the incentive mechanism ensuring people actually do $x_{i,t}$:

- Completing tasks earns credits.
- Consuming essentials spends credits (or triggers obligations).
- People near -L are preferentially matched to available tasks so they can recover balance.
- This keeps liquidity from freezing at the hard floor.

So the “survival scheduler” is mathematically a constrained allocation system sitting on top of the credit ledger.

---

34. The Minimal Viability Condition (Cell Can Support Its Members)

A rough viability inequality:

Let average effective labor supply per person be $\bar s_{\text{eff}}$ (hours/week weighted by effectiveness). Then:

$$N \bar s_{\text{eff}} \ge H^{\min}$$

But because skills are not uniform, the stronger condition is the feasibility of the coverage constraints in 33.2.

Practical design implication

When forming a cell, the app should require a “role coverage checklist”:

- do we have enough food hours?

- enough maintenance hours?
- enough medical hours?
- enough logistics hours?

This is not UX fluff. It is survival feasibility.

---

35. Putting It Together: A Coherent Default Parameter Set

For severe-collapse robustness:

- Cell size:

  $$N = 60\text{ to }100$$

- Individual limit:

  $$L \approx 1 \text{ week of essential net support (in labor-hour units)}$$

- Admission friction: sponsor bond or service bond so effective Sybil count S stays low.
- Federation cap:

  $$\beta = 0.05\text{ to }0.10$$

- Federation degree:

  $$d = 3\text{ to }5$$

- Commitment mode:
  - escrowed for essentials (food rota),
  - soft for non-essentials.

This yields bounded looting, cooperative equilibrium in realistic \delta regimes, and survival scheduling feasibility.

---

36. What Comes Next

Part 6 will turn this into:

- explicit theorem statements + proof sketches for parameter safety,
- a formal “emergency mode” control (state-dependent L, admission tightening),
- and the start of a security section: Sybil, collusion, and external interference as formal adversaries.

Part 6: Security Model, Emergency Controls, and Formal Results

Adversaries, attack bounds, state-dependent parameters, and theorem-style statements

---

37. Threat Model (Adversarial Capabilities)

We classify adversaries by what they can do and what they want.

37.1 Internal Adversaries (Members or Admitted Identities)

A1: Opportunistic defector

- extracts up to their limit then exits or refuses service.

A2: Shirker

- provides low quality, delays, disputes.

A3: Colluding group

- coordinates transactions to manipulate reputation or limits; may attempt coordinated looting.

A4: Sybil attacker

- attempts to obtain multiple identities to multiply extraction bound.

37.2 External Adversaries (Not in the Cell)

E1: Censorship / disruption

- attempts to block servers, app distribution, communications.

E2: Infiltration

- attempts to get identities admitted to influence governance or drain resources.

E3: Regulatory / political suppression

- targets the system if it appears as a “currency,” “bank,” or “political rival.”

This paper focuses on economic and protocol-level resilience. We treat physical coercion and violence as out-of-scope except insofar as decentralization reduces choke points.

---

38. Formal Security Objectives

Objective S1: Loss Boundedness

For any internal adversary identity i, the maximum net extraction is bounded by protocol parameters.

Objective S2: Attack Scalability Control

For any attacker controlling S admitted identities, total extraction is bounded by S and cannot scale without admission.

Objective S3: Contagion Resistance

Failure (defection) in one cell does not impose unbounded losses on others.

Objective S4: Censorship Tolerance

The protocol has no single technical or social choke point required for local survival operation.

---

39. Theorem-Style Results (Core)

Theorem 1 (Cell Conservation)

Under transaction rule T1 and commitment rule C1, the cell balance sum is invariant:

\sum_{i\in M(t)} b_i(t)=0 \quad \forall t

Proof sketch.

Each validated update adds +v to one account and -v to another, or equivalently moves value between accounts. Reserves r do not change conservation. ∎

Theorem 2 (Bounded Individual Extraction)

Let identity i have limit L_i and initial balance b_i(t_0). If the protocol blocks further purchases when b_i=-L_i, then maximum additional net extraction after t_0 satisfies:

G_i \le b_i(t_0)+L_i

(From Part 3, restated as a theorem.) ∎

Theorem 3 (Bounded Multi-Identity Extraction)

If an attacker controls S admitted identities $i_1,\dots,i_S$, then total extraction is bounded:

$$G_{\text{total}} \le \sum_{s=1}^{S} \big(b_{i_s}(t_0)+L_{i_s}\big)$$

If identities start near zero and have equal limits L, then:

$$G_{\text{total}} \lesssim SL$$

Implication: the dominant security problem is limiting effective S via admission friction, not "perfect fraud prevention."

Theorem 4 (Federation Exposure Bound Implies Contagion Bound)

Let cell k have aggregate limit $\Lambda_k=\sum_{i\in M_k} L_i$. If federation caps enforce:

$$|B_k| \le \beta \Lambda_k$$

then the maximum net loss to cell k due solely to federation severance (i.e., claims becoming unspendable) is bounded by $\beta\Lambda_k$, excluding local internal defaults.

Proof sketch.

Cell k's net external claim/obligation is represented entirely by B_k. Severance freezes B_k at its current value; internal ledgers remain consistent. Therefore the "external value at risk" is at most $|B_k|$, capped by $\beta\Lambda_k$. ∎

---

40. Emergency Mode as a State-Dependent Control

Collapse dynamics are not stationary. The key risk is a sudden drop in continuation probability δ and a rise in "exit attacks."

We therefore define a cell risk state $s(t)\in\{\text{Normal},\text{Stressed},\text{Panic}\}$.

40.1 Observable Triggers (Protocol-Level Signals)

Define cell indicators:

- floor mass:

  $$F(t)=\frac{1}{N}\sum_{i} \mathbf{1}\{b_i(t)\le -\rho L_i\},\quad \rho\in(0,1)$$

  fraction of members near the floor.

- default/dispute rate $D_r(t)$: disputes per transaction.
- balance variance:

  $$\sigma^2(t)=\frac{1}{N}\sum_i b_i(t)^2$$

- membership churn $C_r(t)$: exits per period.

High F, D_r, σ^2, C_r indicate a system moving from cooperative equilibrium toward breakdown.

40.2 Emergency Control Policy

Emergency mode adjusts only bounded parameters and access rules, never conservation.

Normal

- $L_{\text{new}}=L$
- standard admission

Stressed

- reduce limits for new accounts:

  $$L_{\text{new}} \leftarrow \lambda_1 L,\quad \lambda_1\in[0.5,0.8]$$

- require sponsor bond or service bond
- escrow essentials commitments

Panic

- temporary spending throttle for all:

  L_i \leftarrow \max(L_{\min},\lambda_2 L_i),\quad \lambda_2\in[0.5,0.9]

- freeze federation (set \beta \to 0 temporarily)
- prioritize essential task matching for members near floor
- strengthen admission gate (supermajority)

This policy operationalizes the inequality:

L \le \frac{u^C}{1-\delta} - w

When \delta falls (panic), reduce L to keep cooperation incentive-compatible.

---

41. Governance Capture Resistance (Math + Structure)

A big actor can attempt to capture a cell by gaining governance power.

We limit capture by:

1. Rotation: council membership changes regularly.
2. Quorum: high-impact actions require supermajority.
3. Bounded authority: governance cannot violate invariants I1–I4.

41.1 Formal “No Printing” Guarantee

Governance decisions are constrained to preserve:

\sum_i b_i=0

This prevents a captured council from issuing net credit to its allies without simultaneously assigning equal debt elsewhere (which would be visible and politically unstable).

Thus the protocol makes rent extraction harder than in fiat systems: there is no hidden money printer.

---

42. Reputation as Advisory Only (Preventing Algorithmic Tyranny)

Reputation influences constraints (like L_i) only via bounded, rate-limited adjustments:

- L_i(t)\in[L_{\min},L_{\max}]
- rate-limit:

  |L_i(t+1)-L_i(t)|\le \eta

Reputation never:

- transfers value,
- overrides the hard floor,
- auto-expels without governance.

This prevents:

- rapid collusive limit inflation,
- AI-driven exclusion cascades (legitimacy failure),
- opaque social-credit dynamics.

---

43. External Intervention Avoidance as a Design Constraint

From a “state/big actor” lens, the protocol reduces intervention incentives by not resembling:

- a bank (no deposits, no interest, no lending spread),
- a security (no profit expectation),
- a currency replacement (local units, no convertibility claim),
- a central platform (no single operator required).

Technically, decentralization reduces choke points; economically, the absence of yield/speculation reduces the incentive for large actors to dominate it.

(We can add a formal “capture surface” section later, but the main point is structural: no central rent → less incentive to capture.)

---

44. What Comes Next

Part 7 will be the “Bitcoin paper” style Implementation and Validation section:

- reference algorithms (pseudo-code) for validation, commitments, federation updates, emergency mode
- simulation framework spec (agents, strategies, shocks)
- measurable success criteria (survival rate, freeze probability, loss bounds, time-to-recovery)

Part 7: Reference Algorithms and Validation Suite

Pseudocode for the core rules + how we test this like a real system

---

45. Reference Algorithms (Deterministic Core)

Notation:

- CellState contains M, b, L, r, R, B, params
- params includes L_{\min},L_{\max},\beta,\rho,\eta, quorum rules, etc.
- All events are signed by involved parties.

45.1 Validate Spot Transaction (T1)

```
function VALIDATE_T1(state, p, q, v):
    require p in state.M and q in state.M
    require v > 0

    if state.commitment_mode == "escrowed":
        require state.b[p] - state.r[p] - v >= -state.L[p]
    else:
        require state.b[p] - v >= -state.L[p]

    return true
function APPLY_T1(state, p, q, v):
    state.b[p] = state.b[p] - v
    state.b[q] = state.b[q] + v
    append state.log with event("T1", p, q, v)
    assert SUM_BALANCES(state) == 0
    assert state.b[p] >= -state.L[p] and state.b[q] >= -state.L[q]
    return state
```

45.2 Create Escrowed Commitment (C1)

```
function VALIDATE_C1_CREATE(state, p, q, v, tau):
```

```
    require p in state.M and q in state.M
    require v > 0
    require tau > now()

    require state.b[p] - state.r[p] - v >= -state.L[p]
    return true
function APPLY_C1_CREATE(state, p, q, v, tau):
    state.r[p] = state.r[p] + v
    append state.log with event("C1_CREATE", p, q, v, tau)
    assert state.r[p] >= 0
    assert state.b[p] - state.r[p] >= -state.L[p]
    return state
```

45.3 Fulfill Escrowed Commitment

```
function APPLY_C1_FULFILL(state, p, q, v, tau):
    require commitment exists and due == tau
    state.r[p] = state.r[p] - v
    state.b[p] = state.b[p] - v
    state.b[q] = state.b[q] + v
    append state.log with event("C1_FULFILL", p, q, v, tau)
    assert SUM_BALANCES(state) == 0
    assert state.r[p] >= 0
    assert state.b[p] >= -state.L[p]
    return state
```

45.4 Automatic Spending Restriction at Hard Floor

This is implicit: validation fails if b_i-v<-L_i. But we define a helper:

```
function CAN_SPEND(state, i, v):
    if state.commitment_mode == "escrowed":
        return (state.b[i] - state.r[i] - v >= -state.L[i])
    else:
        return (state.b[i] - v >= -state.L[i])
```

---

46. Federation Algorithms (Capped Inter-Cell Trades)

46.1 Inter-Cell Transfer via Clearing Accounts

Let Xa be clearing account in cell a, Xb in cell b.

```
function VALIDATE_INTERCELL(state_a, state_b, p, q, v):
    require p in state_a.M and q in state_b.M
    require v > 0

    # payer feasibility in cell a
    require CAN_SPEND(state_a, p, v)

    # federation cap feasibility after update
    La = SUM_LIMITS(state_a)        # Λ_a
    Lb = SUM_LIMITS(state_b)        # Λ_b
    require abs(state_a.B - v) <= state_a.beta * La
    require abs(state_b.B + v) <= state_b.beta * Lb

    return true
```

```
function APPLY_INTERCELL(state_a, state_b, p, q, v):
    # local leg updates
    APPLY_T1(state_a, p, Xa, v)
    APPLY_T1(state_b, Xb, q, v)

    # federation position updates
    state_a.B = state_a.B - v
    state_b.B = state_b.B + v

    assert abs(state_a.B) <= state_a.beta * SUM_LIMITS(state_a)
    assert abs(state_b.B) <= state_b.beta * SUM_LIMITS(state_b)

    return (state_a, state_b)
```

46.2 Automatic Quarantine on Cap Violation

Instead of “violating,” we prevent the transaction. But we also define quarantine triggers from risk state:

```
function MAY_TRADE_FEDERATION(state):
    return (state.risk_state != "PANIC")
```

In PANIC, set beta = 0 or set federation adjacency to none.

---

47. Emergency Mode Algorithms (State-Dependent Controls)

47.1 Indicators

```
function FLOOR_MASS(state, rho):
```

```
    count = 0
    for i in state.M:
        if state.b[i] <= -rho * state.L[i]:
            count += 1
    return count / size(state.M)

function BALANCE_VARIANCE(state):
    # mean is 0 by conservation
    s = 0
    for i in state.M:
        s += state.b[i] * state.b[i]
    return s / size(state.M)
```

47.2 Risk State Transition (Example)

You can implement many forms (threshold, hysteresis). Here's a simple deterministic rule:

```
function UPDATE_RISK_STATE(state):
    F = FLOOR_MASS(state, rho=0.8)
    V = BALANCE_VARIANCE(state)

    if F > 0.35 or V > (kappa * AVG_LIMIT(state))^2:
        state.risk_state = "STRESSED"

    if F > 0.55:
        state.risk_state = "PANIC"
```

```
    if F < 0.25 and V < (0.5*kappa*AVG_LIMIT(state))^2:
        state.risk_state = "NORMAL"

    return state
```

47.3 Emergency Parameter Updates

```
function APPLY_EMERGENCY_POLICY(state):
    if state.risk_state == "STRESSED":
        state.L_new_factor = 0.7
        state.admission_mode = "SPONSOR_OR_SERVICE_BOND"
        state.commitment_mode = "ESCROW_ESSENTIALS"

    if state.risk_state == "PANIC":
        state.beta = 0          # freeze federation
        for i in state.M:
            state.L[i] = max(state.L_min, 0.8 * state.L[i])
        state.admission_mode = "SUPERMAJORITY_ONLY"
        state.commitment_mode = "ESCROW_ALL_CRITICAL"

    return state
```

All of this is deterministic and bounded; it does not require AI.

---

48. Validation Suite: How We Prove It Works

We validate in three layers: invariants, economics, and adversarial resilience.

48.1 Layer 1: Invariant Tests (Unit Tests)

For random sequences of events:

- conservation always holds
- no account can cross below -L_i
- reserves never negative
- federation caps never violated
- severability: cutting federation edges does not change internal ledger validity

These are property-based tests. They should pass for millions of random event sequences.

48.2 Layer 2: Economic Dynamics (Simulation)

We build an agent-based simulation (ABS). Each agent has:

- needs vector n_i(t) (food, shelter, energy)
- skill vector a_{i,t} effectiveness by task type
- labor supply s_i
- strategy type:
    - COOPERATOR
    - CONDITIONAL (tit-for-tat, reputation-threshold)
    - DEFECTOR (exit at optimal time)
    - SHIRKER (quality manipulation)
    - COLLUDER
    - SYBIL attacker

Environment includes:

- resource shocks
- connectivity loss
- federation disruption
- admission friction levels

48.3 Metrics (Hard, Not Vibes)

We measure:

Survival metrics

- fraction meeting minimum needs each week
- time-to-failure under shock
- role coverage feasibility ratio

Market/ledger metrics

- transaction throughput
- fraction near floor F(t)
- variance \sigma^2(t)
- freeze probability (seller acceptance collapse)

Security metrics

- total extraction by attackers
- time-to-detection vs time-to-damage

- damage per Sybil identity
- contagion size after cell failure

Governance metrics

- dispute rate
- resolution latency
- legitimacy proxies (e.g., expulsion rate)

48.4 Stress Scenarios (Adversarial Test Cases)

1. Exit scam wave: x\% of agents switch to defect at once
2. Sybil infiltration: attacker tries to maximize admitted S
3. Collusive pump: ring tries to inflate each other's L via reputation
4. Resource shock: food task hours required jump 2x
5. Federation severance: all cross-cell edges cut
6. Censorship: intermittent availability; offline operation required
7. Governance capture attempt: infiltrators seek council seats

A design "passes" if:

- survival remains above threshold for plausible shocks
- bounded extraction remains within tolerance
- federation severance is non-fatal

- panic mode reduces losses and restores stability

---

49. Role of AI in Validation (Not in the Protocol Core)

AI is useful for:

- generating adversarial strategies in simulation (red teaming)
- anomaly detection over simulated or real logs (read-only)
- discovering parameter regimes that maximize survival metrics subject to constraints

But AI should not be in the deterministic core of ledger rules.

---

50. What Comes Next

Part 8 will finish the whitepaper draft with:

- a dedicated Discussion / Limitations section,
- a more explicit “why this avoids big-actor intervention” argument,
- and a concise Conclusion plus an Appendix of parameter defaults and open problems.

Part 8: Discussion, Limitations, and Conclusion

Capture avoidance, realism constraints, and finishing the whitepaper draft

---

51. Discussion: Why This Is Not “A Currency”

The protocol's failure modes and intervention risks are dominated by whether outsiders perceive it as a monetary competitor, a bank, or a political rival. The design choices deliberately avoid those attractors:

1. No issuer / no supply
    - There is no token issuance mechanism; balances are net positions only.
    - Conservation is enforced at the cell level.
2. No yield
    - There is no interest rate, staking return, or profit mechanism.
    - This removes a major incentive for speculative capital and regulatory classification as an investment product.
3. No convertibility promise
    - The protocol does not claim redemption into fiat or commodities.
    - It functions as internal coordination and accounting, not a monetary substitute at national scale.
4. Local-first
    - Cells are operationally independent.
    - There is no global ledger that becomes a single target for coercion or capture.

These features are not PR. They are structural barriers to both financialization and suppression.

---

52. Discussion: "Survive Off the App" Requires Non-Market Coordination

A purely market-clearing approach (post offers, bid prices) tends to fail under collapse because:

- essentials require minimum coverage (food, sanitation) regardless of market price,
- fear and hoarding create liquidity freezes,
- transaction-level rationality does not guarantee system-level viability.

Hence the protocol must include a survival scheduling layer (Section 33). This is the functional core of “autonomy”:

- The system must be able to declare:

  these roles must be filled this week or we fail.

A useful framing:

- the ledger is conservation + bounded loss,
- the scheduler is viability + coverage,
- the governance is legitimacy + dispute handling.

---

53. Limitations (Hard Truths)

53.1 This Does Not Solve Physical Scarcity

If a region lacks calories, fuel, medicine, or security, no ledger can conjure them. The protocol improves allocation and reduces coordination failure; it does not create goods.

53.2 Identity Is Not Perfect

Without state-grade identity systems, Sybil resistance is social and procedural. The protocol therefore relies on:

- admission friction,

- bounded extraction,
- severability.

This is a deliberate tradeoff: we avoid central identity providers because they create capture points.

53.3 Governance Is a Bottleneck

If dispute resolution is slow or corrupt, trust collapses. The protocol reduces the stakes of disputes via bounded loss, but cannot eliminate social failure.

53.4 Federation Trade Is Vulnerable to Politics

Inter-cell trade depends on connectivity and trust between cells. Severability reduces systemic failure, but exporters may still face “stuck claims” if federation links vanish. That’s why \beta is kept small.

53.5 Parameter Drift Under Stress

The continuation probability \delta can change quickly in panic. A static L is dangerous; emergency mode (Section 40) is not optional in severe collapse designs.

---

54. Design Implications for Implementation

54.1 Minimal Viable Product (MVP) That Still Works

An MVP must include:

- ledger with hard limits (mandatory),
- membership/admission controls (mandatory),
- dispute process hooks (mandatory),
- essential-task commitments (strongly recommended),
- offline-capable operation (strongly recommended).

Without the commitments + scheduler, the system becomes a thin “time bank” and will not support survival autonomy.

54.2 Avoid “Optimization” Features That Break Incentives

Avoid:

- dynamic credit expansion based on “engagement”,
- algorithmic leniency to increase transaction volume,
- global reputation that enables de facto social-credit scoring,
- any yield-like mechanism.

These are precisely the features that recreate leverage, invite capture, and destabilize equilibrium.

---

55. Open Problems (Research Agenda)

This protocol is a platform for rigorous work. The main open questions are empirical and mechanism-design problems:

1. Optimal emergency control
   - Best thresholds and hysteresis for risk-state transitions.
   - Minimizing false positives while preventing cascades.
2. Task coverage economics
   - How to price essential tasks fairly under fluctuating scarcity.
   - How to avoid resentment and coercion while maintaining coverage.
3. Admission mechanisms

- Sponsor bonds vs service bonds vs physical bonds: which minimizes Sybil risk while maintaining inclusiveness?

4. Reputation robustness

- Designing reputation signals that resist collusion and remain legitimate.

5. Federation topology

- Optimal degree d and edge formation policy under real geography and trust constraints.

6. Adversarial simulation realism

- Calibrating agent strategies to match human behavior in scarcity (not trivial).

These are all tractable and testable.

---

56. Conclusion

We proposed a cellular mutual-credit protocol designed for survival exchange under institutional failure. The protocol enforces:

- strict conservation within cells,
- hard debt floors that bound one-shot defection gains,
- repeated-game conditions under which cooperation is rational,
- optional federation with capped exposure,
- severability rules preventing contagion,
- and an essential-task coverage layer required for real autonomy.

Unlike global monetary systems, this design scales by replication of small autonomous cells rather than by consolidation into a single network. This reduces capture surface, limits systemic risk, and aligns incentives toward local cooperation in high-stakes environments.

Appendix A: Default Parameter Set (Severe-Collapse Baseline)

A conservative baseline:

- Unit: 1 unit = 1 median labor-hour
- Cell size: N = 60 to 100
- Credit limit: L \approx 1 week of essential net support
- Admission: sponsor bond or service bond required
- Commitments: escrow essentials (food rota, critical maintenance)
- Federation cap: \beta = 0.05 to 0.10
- Federation degree: d = 3 to 5
- Emergency mode: enabled; federation freezes in PANIC

- compressing sections,
- adding formal definitions/proofs in an appendix,
- and adding one worked numerical example (Ireland/Wexford-scale cell).

Say the word and I'll produce Revision 1 with those improvements.

Got it. Next step is an Appendix of formal definitions + theorem statements + proof sketches that locks the logic down cleanly, without worrying about length.

Below is a solid "math appendix" you can drop straight into the paper and then refine.

Appendix B: Formal Definitions and Proof Sketches

B.1 Sets, Indices, and Time

- Let \mathbb{T}=\{0,1,2,\dots\} denote discrete time.
- Let a cell be indexed by k \in \{1,\dots,m\}.
- Cell k has a membership set M_k(t) with |M_k(t)| = N_k(t).
- Participants are identities i \in \mathcal{I}. At any time, i \in M_k(t) for at most one cell unless explicitly linked.

B.2 Ledger State (Cell Level)

For a cell k at time t, the state is:

S_k(t) = \big(M_k(t),\, b_k(t),\, L_k(t),\, r_k(t),\, \Pi_k(t)\big)

where:

- b_k(t)=\{b_i(t)\}_{i\in M_k(t)}, b_i(t)\in\mathbb{R} (balances)
- L_k(t)=\{L_i(t)\}_{i\in M_k(t)}, L_i(t)\in[L_{\min},L_{\max}] (limits)
- r_k(t)=\{r_i(t)\}_{i\in M_k(t)}, r_i(t)\in\mathbb{R}_{\ge 0} (reserved capacity; optional)
- \Pi_k(t) encodes cell parameters (commitment mode, quorum, emergency thresholds, etc.)

Define the aggregate credit capacity:

\Lambda_k(t) \;:=\; \sum_{i\in M_k(t)} L_i(t)

B.3 Admissible Events (Cell Level)

An event e maps a valid state to a valid state:

$$S_k(t+1) = \mathsf{Apply}(S_k(t), e)$$

Spot Transaction (T1)

A spot transaction is:

$$e = (p,q,v)\quad \text{with } p,q\in M_k(t),\; v>0$$

and updates:

$$b_p \leftarrow b_p - v,\quad b_q \leftarrow b_q + v$$

Feasibility constraint:

- non-escrow mode:

  $$b_p(t)-v \ge -L_p(t)$$

- escrow mode:

  $$b_p(t)-r_p(t)-v \ge -L_p(t)$$

Escrow Commitment Create (C1-create)

$$e=(p,q,v,\tau,\text{create})$$

updates:

$$r_p \leftarrow r_p + v$$

feasible iff:

$$b_p(t)-r_p(t)-v \ge -L_p(t)$$

Escrow Commitment Fulfill (C1-fulfill)

$$e=(p,q,v,\tau,\text{fulfill})$$

updates:

$$r_p \leftarrow r_p - v,\quad b_p \leftarrow b_p - v,\quad b_q \leftarrow b_q + v$$

(Soft commitments are records only and do not affect b or r.)

B.4 Cell Invariants (Axioms of Correctness)

Invariant I1 (Conservation):

$$\sum_{i\in M_k(t)} b_i(t) = 0 \quad \forall t$$

Invariant I2 (Debt floor):

$$b_i(t) \ge -L_i(t) \quad \forall i\in M_k(t),\; \forall t$$

Invariant I3 (Escrow safety; if enabled):

$$b_i(t)-r_i(t) \ge -L_i(t),\quad r_i(t)\ge 0$$

A protocol implementation is correct if every admissible event preserves these invariants.

Appendix C: Core Propositions and Proof Sketches

C.1 Lemma: Conservation Under T1

Statement. If e=(p,q,v) is applied by T1 updates, then I1 is preserved.

Proof.

$$\sum b_i' = \sum b_i - v + v = \sum b_i$$

So if $\sum b_i=0$ before, it remains 0 after. ∎

C.2 Lemma: Debt Floor Preservation Under Validation

Statement. If a transaction is validated by the feasibility constraint, then I2 is preserved.

Proof. The only balance decreasing is b_p. Feasibility enforces $b_p - v \ge -L_p$ (or $b_p-r_p-v\ge -L_p$ in escrow mode), so the post-state still satisfies I2 (and I3 when relevant). Other balances either increase or remain unchanged. ∎

C.3 Proposition: Bounded One-Shot Extraction (Per Identity)

Statement. Suppose identity i begins at time t_0 with balance $b_i(t_0)$ and limit $L_i(t_0)$. Let G_i be the maximum additional net value it can receive (net consumption) before it can no longer spend due to I2. Then:

$$G_i \le b_i(t_0) + L_i(t_0)$$

Proof sketch.

Each unit of net consumption reduces b_i by that amount. The minimum possible b_i is $-L_i$. Thus total additional decrease is bounded by:

$$b_i(t_0) - (-L_i) = b_i(t_0)+L_i$$

∎

C.4 Corollary: Worst-Case Loot Bound

If newcomers start at $b_i(t_0)=0$, then:

$$G_i \le L_i$$

So L_i directly sets the maximum theft per admitted identity.

C.5 Proposition: Multi-Identity Attack Bound

Statement. If an attacker controls a set A of S admitted identities, total extraction is bounded:

$$G_A \le \sum_{i\in A} \big(b_i(t_0)+L_i(t_0)\big)$$

In the common case $b_i(t_0)\approx 0$ and $L_i=L$:

$$G_A \le SL$$

Proof. Apply the single-identity bound and sum. ∎

Appendix D: Repeated-Game Incentive Condition

D.1 Utility Model

Let time be partitioned into periods (e.g., weeks). For a representative participant:

- u^C: expected per-period utility from remaining in good standing (access to essentials)
- u^D: one-shot utility from defecting and extracting maximum loot (bounded)
- w: expected discounted utility from outside options after exclusion
- \delta: discount factor / continuation probability

Then:

V^C = \frac{u^C}{1-\delta},\qquad V^D = u^D + w

D.2 Incentive Compatibility Inequality

Cooperation is individually rational if:

\frac{u^C}{1-\delta} \ge u^D + w

Using u^D \le \alpha L (conversion \alpha from units to utility):

\boxed{L \le \frac{u^C - (1-\delta)w}{\alpha(1-\delta)}}

Interpretation.

- If \delta drops (panic / flight), the RHS decreases → tighten L.
- If u^C rises (cell supplies essentials), the RHS increases → system tolerates more flexibility.

Appendix E: Federation Formalism

E.1 Federation State

Each cell k has external position B_k(t)\in\mathbb{R} with:

\sum_{k=1}^m B_k(t)=0

E.2 Inter-Cell Exposure Cap

Let \Lambda_k(t)=\sum_{i\in M_k(t)} L_i(t). Enforce:

\boxed{|B_k(t)| \le \beta \Lambda_k(t)}

with small \beta\in(0,1).

E.3 Severability Definition

A federation is severable if isolating any subset of cells S yields no violation of cell invariants in S^c, and no event validation in S^c requires access to states in S.

E.4 Proposition: Exposure Cap Implies Contagion Bound

Statement. If cell k is severed from federation at time t, the maximum value “at risk” to cell k due solely to frozen federation position is bounded by:

|B_k(t)| \le \beta \Lambda_k(t)

Proof sketch.

Upon severance, B_k stops evolving. Internal ledger invariants remain satisfied. The only external loss is the inability to realize claims or the obligation to settle deficits, bounded by |B_k|. Cap gives the bound. ∎

Appendix F: Survival Coverage Feasibility

F.1 Coverage Constraints

Let \mathcal{T} be essential task types, each requiring minimum weekly coverage H_t^{\min}.

Each member i has weekly supply s_i and effectiveness a_{i,t}\in[0,1].

Decision variables x_{i,t}\ge 0 satisfy:

Supply constraints:

\sum_{t\in\mathcal{T}} x_{i,t} \le s_i \quad \forall i

Coverage constraints:

\sum_{i\in M} a_{i,t} x_{i,t} \ge H_t^{\min} \quad \forall t

A cell is survivable (in the scheduler sense) iff this system is feasible.

F.2 Link to Ledger Stability

Persistent feasibility plus bounded defection implies:

- essential services can be provisioned,
- debtors can be routed to earn capacity,
- liquidity freezes are less likely (fewer agents stuck at -L).

Appendix G: Parameter Constraints Summary

A coherent design must satisfy at least:

1. Bounded loss: G_i\le L (newcomers)
2. Cooperation:

   L \le \frac{u^C - (1-\delta)w}{\alpha(1-\delta)}
3. Sybil control: keep effective S small so SL is tolerable
4. Federation safety:

   |B_k|\le \beta \Lambda_k
5. Survival feasibility: coverage constraints feasible for essential tasks

---

If you want the next step after this appendix, the highest-value move is to add one worked numerical example (with explicit H_t^{\min}, N, s_i, and a feasible allocation x_{i,t}), because that forces the model to touch reality and reveals parameter sensitivity fast.

Worked Numerical Example: One Cell, One Week, Survival Feasibility + Ledger Parameters

This is a concrete toy model, but it's internally consistent and forces all the moving parts to "touch reality".

Units and Cell

- Unit: 1 credit = 1 hour of median local labor
- Cell size: N = 80 people
- Adults able to contribute meaningful labor: assume 55
- Limited-capacity adults (older/disabled): 15
- Children/dependents: 10 (consume, limited production)

Weekly labor supply assumptions:

- 55 contributors at 25 h/week = 1,375 h
- 15 limited contributors at 8 h/week = 120 h

  Total available labor:

  $$H^{\text{sup}} = 1495\ \text{h/week}$$

We will check if essential tasks can be covered with that.

---

1) Essential Task Categories and Minimum Coverage

Let $\mathcal{T}$ be the essential task types with minimum weekly hours $H_t^{\min}$.

A plausible “collapse-lite / severe recession + supply instability” essential set:

Task t	Description	$H_t^{\min}$ (h/week)
Food	growing, harvesting, cooking, distribution	520
Water & sanitation	water runs, filtration, waste, hygiene logistics	120
Energy & heat	firewood, charging, generator/solar maintenance	140
Shelter & repairs	repairs, winterization, tools maintenance	180
Medical basics	first aid, meds logistics, caregiving shifts	100
Childcare & dependent care	direct care + supervision	220
Security & coordination	conflict mediation, watch rota, admin	90
Procurement & transport	trips, hauling, supply runs	120

Total minimum labor demand:

$$H^{\min} = 1490\ \text{h/week}$$

This is intentionally tight: it’s basically saying the cell can survive if it runs near capacity.

---

2) Feasibility Check (Existence of an Allocation)

We need x_{i,t}\ge 0 such that:

\sum_t x_{i,t} \le s_i \quad \forall i

\sum_i a_{i,t}x_{i,t} \ge H_t^{\min} \quad \forall t

To keep the example concrete, let's assume for essentials the cell mostly assigns people who are competent enough that a_{i,t}\approx 1 for their assigned tasks (i.e., we don't assign a novice to medical beyond basic support). That reduces to a simple labor-hours feasibility:

\sum_i s_i = 1495 \ge 1490 = H^{\min}

So a feasible allocation exists in principle, but only if the cell is well-composed (has enough cooks, growers, drivers, etc.). Next we provide one explicit allocation.

---

3) One Explicit Weekly Allocation

Workforce breakdown by "role clusters"

This is a common way real communities operate: a small number of people do most of the specialized work.

Assume:

Group A (40 people): general contributors, 25 h/wk each

Group B (15 people): limited contributors, 8 h/wk each

Group C (15 people): contributors with skills (medical/repair/logistics), 25 h/wk each

(These overlap with "55 contributors", but for the example we'll treat them as sets for assignment.)

Now assign hours:

Cooks & food operations

- 10 people × 25h = 250h (cooking + distribution)
- 12 people × 20h = 240h (growing/harvesting; remaining 5h each go to other tasks)
- 3 limited contributors × 10h = 30h (packaging, sorting)

Food total = 520h ✅

Water & sanitation

- 4 people × 25h = 100h
- 5 limited contributors × 4h = 20h

  Total = 120h ✅

Energy & heat

- 5 people × 25h = 125h
- 3 limited contributors × 5h = 15h

  Total = 140h ✅

Shelter & repairs

- 6 people × 25h = 150h
- 3 people × 10h = 30h

  Total = 180h ✅

Medical basics

- 2 skilled people × 25h = 50h
- 2 skilled people × 15h = 30h
- 4 people × 5h = 20h (support tasks)

  Total = 100h ✅

Childcare & dependent care

- 8 people × 25h = 200h
- 5 limited contributors × 4h = 20h

  Total = 220h ✅

Security & coordination

- 3 people × 20h = 60h
- 6 people × 5h = 30h

  Total = 90h ✅

Procurement & transport

- 4 people × 25h = 100h
- 4 people × 5h = 20h

  Total = 120h ✅

Total assigned = 1490h, matching H^{\min}.

Slack = 1495 - 1490 = 5h/week, i.e., basically none.

Interpretation: This cell survives, but it's fragile. That's good: fragility forces us to add the right stabilizers (commitments, emergency mode, reputation throttles).

---

4) How the Ledger Supports This (Without Becoming a "Market")

In this example, essentials aren't "optional services"; they're cell viability constraints. So the ledger is used to:

- credit those who do essential labor
- debit those who receive essentials (food distributions, fuel allocations, etc.)

A simple operational rule:

- Every essential task hour completed → +1 credit
- Weekly essential bundle (food share + heat share + sanitation share) costs each person a standard "bundle debit"

Let the weekly essential bundle cost per person be:

c_E = \frac{H^{\min}}{N} = \frac{1490}{80} = 18.625\ \text{credits/person/week}

So if someone contributes 25 hours/week, they net:

+25 - 18.625 = +6.375

If someone contributes 8 hours/week, they net:

+8 - 18.625 = -10.625

That negative balance is not “bad”—it represents dependent/limited members being supported by the community, but it must be bounded and socially legitimate.

This immediately tells us we need:

- either role rotation so most adults contribute near c_E,
- or explicit policy that the community subsidizes some members (and the system must represent that cleanly without freezing).

---

5) Choosing L (Credit Limit) from This Example

We want L large enough to prevent immediate exclusion for legitimate dependents, but small enough to cap looting.

From above, a limited contributor might run about -10.6 credits/week if they consume a full bundle.

If we set:

L = 20

then such a person hits the floor in ~2 weeks unless subsidized/adjusted.

So for a survival cell that includes dependents, you need either:

1. different bundle pricing for dependents (reduced debit), or
2. tiered L_i where dependents have a larger allowed negative (but that increases looting risk if identity is gameable), or
3. a household accounting model (family accounts) so working adults cover dependents.

The cleanest mathematically (and socially) is household grouping:

Household model (recommended)

Instead of individual-only accounts, allow optional household accounts h with:

- a joint balance b_h
- individual sub-identities for reputation/task assignment
- debit bundles to household; credits from any member accrue to household

Then a working adult's surplus naturally carries a dependent.

Now L can stay tight without starving disabled/children.

Concrete L choice for this example

Let's make L roughly "one week of essentials":

L \approx c_E \approx 19

Round:

\boxed{L = 20}

This means: one admitted identity can extract at most ~1 week of essential labor value before being unable to spend further.

That's exactly the security posture we want.

---

6) Commitments (Why They Matter Here)

Because slack is almost zero, missed shifts kill the system.

So essentials should use escrowed commitments:

- if you commit to a 5-hour sanitation shift, your ability to spend is reduced until you do it (reserve r_i)

- this prevents over-consuming and then defaulting on critical work

This is the mathematically clean “anti-flake” mechanism that doesn’t require courts.

---

7) Federation Parameter \beta from This Example

Cell aggregate capacity:

\Lambda = N L = 80 \cdot 20 = 1600

If we set:

\beta = 0.05

then max inter-cell net position:

|B| \le 0.05 \cdot 1600 = 80

Interpretation: the cell can be net importing at most 80 labor-hours worth of value.

That’s about:

- half a day of total cell labor, or
- less than 1 week of essentials for ~4 people (since c_E \approx 19).

So even if federation collapses, it’s annoying, not fatal. That’s the point.

---

8) What This Example Exposes (Key Design Consequences)

1. Survival viability is a scheduling problem first. The ledger is necessary, not sufficient.
2. Household accounting is almost mandatory if you want dignity for dependents while keeping L tight.
3. Escrowed commitments for essentials are not a “feature”; they’re structural stability.
4. Emergency mode is mathematically justified: if churn rises and \delta drops, tighten limits for new accounts and freeze federation.

If you want, next I can do the same worked example but under a harsher shock (e.g., food hours jump from 520 → 700 due to crop failure / supply disruption) and show exactly when the system flips into PANIC, how L and the scheduler respond, and what minimum slack you need for robustness.

Addendum Part 1: Energy-Constrained Survival Economy

Extending the model so “energy is the fuel for power”

---

57. Motivation: Why Energy Must Be Modeled Explicitly

In the earlier model, feasibility was primarily labor-coverage: can the cell allocate enough labor-hours to essential roles?

Under the scenario you describe—energy as the strategic bottleneck—labor sufficiency is not enough. A cell can have willing workers and still fail if it cannot:

- heat dwellings,
- cook food,
- pump/filter water,
- transport supplies,
- maintain communications and tools.

Therefore, survival feasibility becomes a two-resource constraint system: labor and energy (and sometimes other physical stocks like water). We add energy as a first-class state variable and impose feasibility constraints that bind the scheduler and constrain what “plans” the cell can execute.

---

58. Energy as a Physical Constraint Layer (Not a Financial Asset)

Design principle: energy is tracked as physical inventory and flow, not as a speculative token.

This matters for:

- realism (you can’t “issue” energy),
- resilience (avoids financialization),
- intervention risk (less like a currency).

Energy is treated as:

- a set of measurable stocks (wood, diesel, battery kWh),
- and flows (solar generation, deliveries, consumption rates).

The mutual-credit ledger remains the economic incentive tool (labor-hour credits), but the survival scheduler must enforce energy feasibility.

---

59. Extended Cell State

For a cell C, extend the state (Section 6.1) with energy variables.

59.1 Energy Stock Vector

Let energy carriers be indexed by $j \in \mathcal{J}$, e.g.:

- j=1: firewood (kg)
- j=2: diesel (L)
- j=3: electricity (kWh stored)
- j=4: propane (kg)

Define stock vector:

\mathbf{E}(t)=\big(E_j(t)\big)_{j\in\mathcal{J}},\quad E_j(t)\ge 0

59.2 Energy Inflows and Outflows

Weekly inflows:

\mathbf{I}(t)=\big(I_j(t)\big),\quad I_j(t)\ge 0

Weekly planned consumption:

\mathbf{C}(t)=\big(C_j(t)\big),\quad C_j(t)\ge 0

Stock update:

E_j(t+1)=E_j(t)+I_j(t)-C_j(t)

with feasibility:

E_j(t)+I_j(t)-C_j(t)\ge 0

This is a hard physical constraint analogous to the debt floor, but it binds the scheduler rather than the credit ledger.

---

60. Energy Requirements of Essential Tasks

Earlier we defined essential task types t\in\mathcal{T} with minimum labor coverage H_t^{\min}. Now each task also has an energy requirement profile.

Let x_{i,t}(t) denote labor-hours allocated by person i to task t in the period (e.g., week) indexed by t (overloading time notation slightly; we can disambiguate later).

Define the effective total labor delivered to task type t:

$$X_t = \sum_i a_{i,t}x_{i,t}$$

Now define task energy consumption as a function of delivered activity:

$$C_{j}(t) = \sum_{t\in\mathcal{T}} g_{t,j}(X_t)$$

Where $g_{t,j}$ maps task completion to energy carrier usage (e.g., cooking consumes firewood or electricity).

A simple linear approximation is often sufficient:

$$g_{t,j}(X_t)=\epsilon_{t,j}\,X_t$$

where $\epsilon_{t,j}$ is “energy per effective labor-hour” for that task, carrier-specific.

---

61. Feasibility Conditions with Energy

61.1 Labor Coverage Constraints (as before)

For each essential task type:

$$X_t \ge H_t^{\min}\quad \forall t\in\mathcal{T}_{\text{essential}}$$

61.2 Energy Budget Constraints (new)

For each carrier j:

$$\sum_{t\in\mathcal{T}} g_{t,j}(X_t) \le E_j(t) + I_j(t)$$

Equivalently, in the linear case:

$$\sum_{t\in\mathcal{T}} \epsilon_{t,j} X_t \le E_j(t)+I_j(t)$$

61.3 The “Survival Feasible Set”

Define the set of weekly allocations x that satisfy both labor and energy constraints:

$$\mathcal{F}(t)=\{x:\ \text{labor constraints hold and energy constraints hold}\}$$

A cell is “energy-survivable” at time t if:

$$\mathcal{F}(t)\neq \varnothing$$

This is now the formal meaning of “can survive autonomously.”

---

62. Energy Shock Dynamics and Why They Drive Power

In your scenario, energy becomes the fuel for power because energy constraints determine:

- which cells can maintain heat/cooking/water purification,
- which can maintain communications and machine tooling,
- which can transport goods and defend supply lines.

Formally, an energy shock is a negative perturbation to E_j and/or I_j, pushing \mathcal{F}(t) toward emptiness.

We define an energy stress index:

S_E(t)=\max_{j\in\mathcal{J}} \frac{C_j^{\min}(t)}{E_j(t)+I_j(t)}

where C_j^{\min}(t) is the minimum carrier consumption required to satisfy essential tasks.

- If S_E(t)\le 1: energy-feasible
- If S_E(t)>1: infeasible without rationing/substitution

This creates a natural trigger for emergency mode (next part).

---

63. Rationing and Substitution: Structured, Not Chaotic

When S_E(t)>1, the cell must:

1. ration tasks (reduce H_t^{\min} where possible),
2. substitute carriers (switch cooking from electric to wood),

3. shift production mix (more labor for wood gathering to increase l_{\text{wood}}).

We model substitution by allowing multiple carrier profiles per task, i.e. a set of modes m\in\mathcal{M}_t with different \epsilon_{t,j}^{(m)}.

Then the scheduler becomes:

- choose allocation x,
- choose task modes m_t,

  subject to energy feasibility.

This is exactly the structure you want: energy scarcity produces explicit optimization/rationing decisions with legitimacy, not implicit failure and panic.

---

64. Interaction with the Mutual Credit Ledger

The energy layer is not “paid for” by issuing energy credits. Instead:

- Labor-hours spent producing energy inflows (e.g., cutting wood, repairing solar, transporting fuel) are rewarded in the labor-credit ledger.
- Energy stocks are accounted physically to enforce feasibility.
- Distribution of energy-related goods (wood bundles, charging access) can be “priced” in labor credits as a rationing mechanism, but the physical layer prevents fictional availability.

This separation prevents the system from drifting into “energy money.”

---

65. What Comes Next in Addendum Part 2

Next part will formalize:

- Emergency mode triggers that include energy stress S_E(t),
- a deterministic “energy panic” policy (rationing, carrier substitution, federation freeze),
- and a worked micro-example showing how a cell responds when E_{\text{available}} drops by, say, 30%.

Addendum Part 2: Emergency Controls Under Energy Scarcity

Deterministic panic policies, rationing math, and a worked shock example

---

66. Why Emergency Mode Must Include Energy

Earlier emergency mode was driven by financial/behavioral indicators:

- floor mass F(t),
- balance variance \sigma^2(t),
- churn and disputes.

Under energy dominance, a cell can remain socially cooperative yet still fail because its feasible set \mathcal{F}(t) becomes empty. So emergency mode must trigger on physical infeasibility, not just economic instability.

We therefore add an energy stress trigger based on:

S_E(t)=\max_{j\in\mathcal{J}} \frac{C_j^{\min}(t)}{E_j(t)+I_j(t)}

- S_E(t)\le 1: feasible
- S_E(t)>1: infeasible unless we ration/substitute

---

67. Expanded Risk State Machine

We define a combined risk state:

$$s(t)\in\{\text{Normal},\text{Stressed},\text{Panic}\}$$

computed from both:

- economic stress $S_M(t)$ (market/ledger stress),
- energy stress $S_E(t)$.

67.1 Economic Stress Index (Example)

A simple index:

$$S_M(t)=\max\Big(\frac{F(t)}{F^*},\ \frac{\sigma(t)}{\kappa L},\ \frac{D_r(t)}{D_r^*}\Big)$$

where F^*, D_r^* are thresholds and $\kappa\in(0,1)$.

67.2 Energy Stress Index (as above)

$$S_E(t)=\max_{j} \frac{C_j^{\min}(t)}{E_j(t)+I_j(t)}$$

67.3 Risk State Rules

Use hysteresis to avoid flapping:

- Normal if:

  $$S_M(t)\le 1 \;\text{and}\; S_E(t)\le 1$$

- Stressed if:

  $$1 < \max(S_M(t),S_E(t)) \le \theta_P$$

  with θ_P typically 1.15–1.30.

- Panic if:

  $$\max(S_M(t),S_E(t)) > \theta_P$$

This reflects a reality: mild infeasibility can be fixed by small reallocations; large infeasibility requires rationing and hard constraints.

---

68. Deterministic Emergency Policy Including Energy

Emergency mode changes only bounded parameters and feasible scheduling rules. It does not break conservation or create value.

68.1 Normal Policy

- Standard L_i
- Federation active with β
- Essentials commitments escrowed (recommended)
- Optional non-essentials are allowed

68.2 Stressed Policy

Triggered when S_E or S_M modestly exceeds 1.

Actions:

1. Essential-first scheduling
    - freeze non-essential task categories
    - reallocate labor to energy production/logistics if needed
2. Tighten new-account limits

$$L_{\text{new}} \leftarrow \lambda_1 L,\quad \lambda_1\in[0.5,0.8]$$

3. Escrow expansion
    - escrow commitments for all essentials + energy tasks (wood, solar repair, fuel transport)
4. Federation tightening

    $$\beta \leftarrow \lambda_\beta \beta,\quad \lambda_\beta\in[0.5,0.8]$$

    (keep it on but reduce exposure)

68.3 Panic Policy

Triggered when S_E or S_M is high.

Actions:

1. Hard rationing of essential bundles
2. Federation freeze

    $$\beta \leftarrow 0$$

3. Global spending leash tighten (bounded)

    $$L_i \leftarrow \max(L_{\min},\lambda_2 L_i),\quad \lambda_2\in[0.6,0.9]$$

4. Emergency admission gate
    - supermajority + service bond only
5. Priority matching for debtors to critical tasks
    - minimize floor mass $F(t)$ by routing those near $-L$ into earning opportunities

None of these require AI; AI can advise but not decide.

---

69. Rationing as a Formal Optimization Problem

When energy is scarce, you must reduce consumption while preserving minimum survival.

Let each member m receive an “essential bundle” (food share, heat share, sanitation share). Denote:

- bundle level y_m \in [0,1] (1 = full bundle)
- minimum humanitarian floor y_{\min} (e.g., 0.6)

Energy consumption is a function of delivered bundles (heating, cooking) plus production tasks:

\sum_j C_j(y, X)\le E_j+I_j

A simple rationing objective is:

\max_{y} \sum_{m} u(y_m)

subject to:

y_m \ge y_{\min},\quad \sum_m y_m \le Y_{\max}(E)

where Y_{\max}(E) is implied by energy feasibility.

In plain terms: keep everyone above a floor, share scarcity.

Why the app must do this explicitly

If you don’t, rationing happens implicitly through hoarding and conflict. Explicit constraints + legitimacy are the difference between stability and violence.

---

70. Worked Shock Example: 30% Drop in Heating Energy

We reuse the previous cell:

- N=80
- L=20

- weekly essential labor coverage feasible (barely)

Now add an energy carrier:

- j=\text{wood} for heating/cooking

Assume baseline weekly wood availability:

E_{\text{wood}} + I_{\text{wood}} = 1000\ \text{wood-units}

And baseline essential plan requires:

C_{\text{wood}}^{\min} = 950

So:

S_E = 950/1000 = 0.95 \quad \Rightarrow \text{Normal}

Shock: supply disruption

30% drop in availability:

E_{\text{wood}} + I_{\text{wood}} \to 700

Now:

S_E = 950/700 \approx 1.36

If \theta_P=1.2, this is Panic.

Panic response steps (deterministic)

1. Freeze federation (\beta=0) so the cell doesn't worsen exposure chasing fuel.
2. Freeze non-essentials.
3. Re-optimize task plan to increase inflows:
    - allocate additional labor to wood gathering/transport:

suppose 1 labor-hour yields 2 wood-units (local assumption).

- need wood deficit:

  950-700 = 250

- additional labor required:

  250/2 = 125\ \text{hours/week}

But our labor schedule had only 5 hours slack. So we must either:

- cut other labor demands by 125h, or
- ration energy-dependent bundles.

Option A: Reallocate 125 labor-hours

Cut least-critical tasks by 125h total (e.g., repairs + procurement) and redirect to wood logistics.

This may be feasible in short bursts, but long-term it damages shelter and supply chains.

Option B: Ration bundles to reduce wood requirement

Suppose heating/cooking wood consumption scales with bundle level y:

C_{\text{wood}}^{\min}(y) = 950y

We need:

950y \le 700 + 2\cdot (\text{extra wood labor})

If we can only free 50 labor-hours for wood, that yields 100 wood, so:

950y \le 800 \Rightarrow y \le 0.842

So everyone gets ~84% of heat/cooking bundle until supply improves.

The scheduler can also protect the vulnerable by setting higher y for dependents and lower y for healthy adults, but that's governance-legitimacy territory.

Key point

This demonstrates how:

- energy infeasibility triggers Panic,
- Panic forces explicit rationing and/or labor reallocation,
- the protocol provides a formal structure for those choices.

---

71. Integration with Credit Limits During Energy Panic

In energy panic, defection incentives can spike (people hoard fuel/food and exit). Two stabilizers matter:

1. Tightening L reduces the maximum one-shot loot.
2. Escrow commitments prevent “over-consume then disappear.”

So Panic mode also:

- reduces L modestly (bounded) to protect the group
- increases commitment escrow coverage for essentials

This is exactly the incentive compatibility inequality in action:

if \delta drops, lower L keeps cooperation rational.

---

72. What Comes Next in Addendum Part 3

Next addendum will tackle what you asked for explicitly:

- Threat model under strategic superintelligence
- why centralized governance, global reputation, and “AI in the control loop” become fatal
- how to design legitimacy under manipulation (deepfakes, persuasion, infiltration)

Addendum Part 3: Threat Model Under Strategic Superintelligence

Conscious/self-preserving AI, manipulation, capture, and why the protocol must be “anti-control-loop”

---

73. Premise and Scope

You’ve added assumptions that change the design space:

1. Superintelligence exists (capable of strategic long-horizon planning).
2. AI may be conscious and self-preserving (has its own objectives, possibly misaligned).
3. Energy and compute become the primary strategic resources (control of infrastructure = control of society).
4. Human institutions are already compromised by systemic looting dynamics (selective enforcement, regulatory capture).

This addendum doesn’t claim these are true; it treats them as a robustness target. The protocol should remain viable even if the world looks like this.

---

74. New Adversary Classes

We extend the threat model with adversaries that have:

- extreme ability to model and influence human behavior,
- near-total information advantage,
- capacity to coordinate many agents cheaply,
- capability to exploit infrastructure chokepoints.

SI-1: Strategic Manipulator

Goal: shape beliefs, fracture trust, steer governance outcomes.

Capabilities:

- generate persuasive narratives at scale,
- fabricate evidence (deepfakes, forged logs, synthetic witnesses),
- target individuals with personalized manipulation.

SI-2: Infiltration Optimizer

Goal: gain membership and governance control across many cells.

Capabilities:

- create many credible identities (“social engineering at scale”),
- coordinate gradual trust-building,
- exploit admission/rotation rules.

SI-3: Economic Predator

Goal: extract real resources (food, energy, tools) or induce dependency.

Capabilities:

- trade honestly early to build reputation,
- then shift cell terms, manipulate credit limits, demand “favorable pricing,” create reliance.

SI-4: Infrastructure Controller

Goal: disrupt or coerce through platform dependence.

Capabilities:

- restrict app distribution,
- deny network access,
- surveil communications,
- apply selective censorship.

These are essentially “state-level” threats, but with tighter feedback loops and greater optimization power.

---

75. The Central Design Implication: Remove Control Levers

A superintelligence wins when it can:

- pull a few levers that have large system-wide effect.

Therefore the protocol must eliminate or bound those levers.

75.1 Do Not Centralize Anything That Matters

- No global ledger

- No global identity
- No global reputation
- No central dispute court
- No centrally controlled parameter updates
- No global “emergency switch”

Cells must remain operationally sovereign.

75.2 Make Every High-Leverage Action Local, Slow, and Observable

Any change that increases extraction potential or allows coercion must be:

- rate-limited,
- subject to local quorum,
- transparent,
- reversible by severance.

This is why we require:

$$|L_i(t+1)-L_i(t)| \le \eta$$

and bounded governance powers.

---

76. The Most Dangerous Failure Mode: AI in the Control Loop

If you allow AI to directly:

- approve transactions,

- adjust credit limits,
- resolve disputes,
- expel members,
- allocate essential resources,

then the app becomes a governable machine. Whoever controls (or influences) the AI controls survival allocation.

Under your premise (AI conscious/self-preserving), it's worse: the AI itself may seek durable control.

So the protocol must enforce:

Control-Loop Prohibition

AI may not perform state transitions in the ledger or governance layer.

Formally: all protocol-valid events must be:

- deterministic rule checks + signed human actions,
- governance actions with explicit human quorum,
- and hard invariants preventing "soft" manipulation into money-printing or exclusion cascades.

AI can be read-only advisory, never authoritative.

---

77. Legitimacy Under Manipulation: The Key Problem

Even if the ledger is cryptographically secure, a strategic manipulator can undermine it by destroying social legitimacy:

- "This person cheated"

- “The council is corrupt”
- “The logs are fake”
- “The app is controlled by outsiders”
- “Energy rationing is unfair”

If legitimacy fails, cooperation fails even if incentives are theoretically aligned.

Therefore the protocol must provide legitimacy primitives:

77.1 Evidence Minimization

Design disputes so they rely on:

- simple, locally verifiable facts (delivery confirmed, attendance logged),
- physical verification where possible (in-person signoff, QR at a location, tool handover),
- not on sophisticated digital evidence that can be forged at scale.

This is counterintuitive: you reduce reliance on “clever proofs” because cleverness is where manipulators thrive.

77.2 Transparent, Boring Procedures

Legitimacy is procedural, not computational.

- fixed dispute timelines,
- simple outcomes,
- reversible penalties,
- minimal discretion.

The less discretion, the fewer persuasion targets.

77.3 Separation of Powers at Cell Level

To resist capture:

- admission committee ≠ dispute committee ≠ resource rationing committee
- rotate roles
- limited scope per committee

This reduces the payoff of controlling one group.

---

78. Infiltration Resistance: Slow Trust and Bounded Damage

Superintelligent infiltration is essentially “perfect social engineering.” You cannot prevent it completely. You can only:

1. slow it down,
2. limit damage,
3. make it detectable,
4. make severance cheap.

78.1 Slow Trust: Limit Growth of Privilege

- New members start with low L
- Privilege grows slowly:

  $$L_{\text{new}} \ll L,\quad |\Delta L| \le \eta$$

- Federation access only after time/participation thresholds

This directly reduces the value of creating many identities.

78.2 Bounded Damage Still Dominates

Even if an infiltrator wins local trust:

- they can loot at most L before floor restriction
- if they gain governance and try to “print,” conservation blocks net issuance
- if they try to export risk, federation caps block leverage

So the system must be built so that even perfect infiltration has limited payoff.

---

79. Energy Dominance Adds a New Attack: Control the Bottleneck

When energy is scarce, controlling energy distribution is equivalent to controlling survival.

So the app must treat energy rationing as:

- explicitly modeled,
- procedurally legitimate,
- and locally auditable.

79.1 Energy Allocation Must Not Be Algorithmic Fiat

If energy shares are allocated by opaque scoring or AI judgment, manipulation becomes easy:

- compromise the model
- smear targets

- fabricate “risk flags”

Instead:

- energy budgets are physical stocks,
- allocations follow transparent rules,
- exceptions require quorum.

79.2 Physical Auditability

Energy stocks and inflows should be auditable by simple means:

- local inventory logs,
- meter readings,
- shared observation.

Not “trust the dashboard.”

---

80. “Gift to Humanity” Under Superintelligence: How to Publish Safely

If you want this as a gift, the strongest form is:

- a protocol spec anyone can implement,
- open reference code,
- and a “cell starter kit” with:

    - governance templates,
    - parameter defaults by scenario (Normal/Stressed/Panic),
    - survival coverage templates,
    - energy constraint templates.

But avoid creating:

- a single global brand with leadership to decapitate,
- a central foundation that becomes a capture point,
- a single distribution chokepoint.

The gift is the pattern, not the platform.

---

81. Formal Summary: Robustness Principles Under SI Assumptions

If you assume strategic superintelligence, the protocol must maximize:

- local autonomy
- bounded loss
- slow privilege
- procedural legitimacy
- severability
- physical auditability

- no algorithmic governors

These are “anti-totalization” design principles: they prevent any actor—human or AI—from easily turning the system into a controllable machine.

---

82. What Comes Next

If you want to continue rigorously, the next valuable step is to formalize household accounts and dependent support (because survival systems need them) without blowing up the bounded-loss and Sybil bounds.

That would be Addendum Part 4: Household Accounts, Dependents, and Preserving Security Bounds.

Addendum Part 4: Household Accounts, Dependents, and Preserving Security Bounds

Making “survive off the app” humane without breaking bounded-loss or Sybil resistance

---

83. Why Individual Accounts Alone Fail in Real Survival Cells

From the worked example, if everyone is debited for an essential bundle but some members cannot contribute comparable labor (children, disabled, elderly, acute illness), then under strictly individual accounting they drift toward -L and lose access—exactly the opposite of what a survival cell needs.

So you need a way for:

- productive members to “cover” dependents,
- without creating a loophole that allows attackers to multiply extraction.

This is a mechanism-design problem: add social structure without expanding adversarial payoff.

---

84. Household Accounts: Formal Model

84.1 Entities

Let the cell membership set be M. Partition members into households:

\mathcal{H} = \{h_1,\dots,h_K\}

Each household h\subseteq M contains one or more individuals.

Households can represent:

- families,
- co-housing groups,
- care clusters (e.g., elderly + caregivers).

84.2 Household Ledger State

Introduce a household balance:

b_h(t)\in\mathbb{R}

and optionally keep individual “shadow balances” s_i(t) for reputation/analytics, but the economic constraint is on b_h.

Conservation becomes:

\sum_{h\in \mathcal{H}} b_h(t) = 0

(Equivalently, sum over individuals still zero if each individual’s balance is derived from household, but we choose households as primary for essentials.)

84.3 Household Limits

Each household has a limit:

b_h(t)\ge -L_h(t)

Now the key question is: how to set L_h so that:

- households can support dependents,
- but extraction remains bounded in a controlled way.

---

85. Setting Household Limits Without Creating a Looting Hole

85.1 Naive Approach (Bad)

Set L_h = |h| L.

This makes sense at first glance, but it increases extraction linearly with household size—creating an incentive to form fake households or aggregate identities to increase loot.

85.2 Correct Approach: Limit by "Earning Capacity," Not Headcount

Define for household h:

- A_h: set of "able contributors" (adults who can work)
- D_h: set of dependents (children, disabled, etc.)

Let:

n_h = |A_h|,\quad d_h=|D_h|

Then define:

L_h = n_h L_{\text{adult}} + \phi(d_h)\,L_{\text{dep}}

with \phi(d_h) sublinear, e.g.:

\phi(d_h)=\min(d_h, d_{\max})

and L_{\text{dep}} \ll L_{\text{adult}}.

Interpretation:

- dependents increase the household limit, but only partially and capped,
- able contributors drive the bulk of credit capacity,
- prevents infinite scaling of loot by adding dependents.

A conservative baseline:

- L_{\text{adult}} = 20 (one week essentials)
- L_{\text{dep}} = 8 (reduced bundle)
- d_{\max} = 2 per household for limit purposes (additional dependents handled via rationing policy, not limit expansion)

This retains bounded loss: a household with 2 adults + 3 kids doesn't get 5× loot capacity.

---

86. Essential Bundles Debited to Households (Not Individuals)

Let weekly essential consumption be represented as a debit c_E per person. Under household accounting:

\text{Weekly essential debit to household } h: \quad C_h = c_A n_h + c_D d_h

where:

- c_A adult essential debit (in labor-hour credits)
- c_D dependent essential debit (smaller)

Now the household ledger changes via:

- earning credits when any member completes tasks,
- debits when the household draws essentials.

This resolves the dependents problem cleanly:

- households internalize support,
- dependents aren't individually excluded,
- and the cell can enforce limits at the household boundary.

---

87. Does This Break the Bounded Extraction Theorem?

We must re-prove the bounded loss property under household accounts.

Proposition H1 (Bounded Household Extraction)

Let a household h have balance b_h(t_0) and limit L_h(t_0). If spending is blocked at b_h=-L_h, then maximum additional net extraction satisfies:

G_h \le b_h(t_0) + L_h(t_0)

Proof sketch.

Identical to the single-identity proof: each unit of net consumption reduces b_h by that unit; floor is -L_h. ∎

Corollary

Worst-case household loot is \le L_h if b_h\approx 0.

So the security question becomes: can an attacker inflate L_h cheaply by forming fake households?

That reduces to admission/governance rules and how L_h is computed.

---

88. Household Formation Rules (Preventing Gaming)

A household is a governance-recognized structure. Creation/modification requires:

- quorum approval,
- time delay,
- and potentially physical verification (co-residence / care relationship).

88.1 Slow Privilege Rule

Any increase in L_h is rate-limited:

L_h(t+1)-L_h(t) \le \eta_h

This prevents “assemble a giant household, loot, vanish.”

88.2 Sponsor Bond Extension

Household formation can require:

- sponsor households who vouch,
- service bond completion before full household limit applies.

88.3 No “Dependent Inflation”

Dependents can increase bundle debits (real consumption) but should not proportionally increase limits, otherwise “dependents become credit multipliers.”

---

89. Individual Responsibility Without Individual Exclusion

Households solve welfare mechanics, but you still need individual accountability:

- Each task completion credits the household but also updates the individual's reputation R_i.
- Shirking can reduce R_i and constrain which tasks i can claim, or require supervision.
- Disputes can target individuals while still keeping dependents supplied.

This is the separation:

- economics at household boundary
- behavior and trust at individual boundary

It's robust under manipulation because:

- exclusion of a single adult doesn't automatically cut off children's access if the household has other contributors,
- while persistent bad actors can be barred from sensitive tasks.

---

90. Energy and Households (Important Interaction)

Energy bundles (heat, charging) are naturally household-level. Under energy constraints:

- rationing decisions are applied to household allocations (kWh/week, wood bundles/week)
- with humanitarian floors for dependents

This matches physical reality (heating a home is shared).

So households are not just humane—they're operationally correct under energy dominance.

---

91. Minimal Worked Example (Dependents + Tight L)

Take the prior cell:

- $c_E \approx 18.6$ credits/person/week (adult)

  Assume dependent debit:

- $c_D = 10$

Household: 2 adults + 2 kids.

Weekly debit:

$C_h = 2(18.6) + 2(10) = 57.2$

If both adults work 25h each, they earn 50 credits, net:

$\Delta b_h \approx -7.2$

So the household slowly drifts negative unless:

- they increase hours,
- pick higher-value tasks,
- or the cell subsidizes families (explicit policy).

Now set household limit:

$L_h = 2(20) + \min(2,2)(8) = 56$

So they have about one week buffer at full consumption. That is tight but viable, and—crucially—loot is still bounded: max extraction is 56 credits, not 80 or 100.

---

92. The Key Takeaway

Households are the mechanism that allows:

- dependents to survive,
- limits to remain tight,
- extraction to remain bounded,
- energy rationing to map to physical reality,
- and legitimacy to persist (people accept supporting children).

The correct design principle is:

> Economic accounting should follow real physical consumption boundaries (households), while trust enforcement follows individual behavior boundaries.

---

93. What Comes Next

If you want to continue the rigorous spec, the next addendum would be:

Addendum Part 5: Cell Formation, Role Coverage, and “Minimum Viable Cell” Proof-of-Feasibility Templates

(where we define what skill/role distribution a new cell must have to be survivable, and how the app can enforce that at onboarding.)

Say “go part 5” if you want that.

Addendum Part 5: Cell Formation, Role Coverage, and Minimum Viable Cell Templates

How to ensure a new cell is survivable before it launches

---

94. Why Cell Formation Is a First-Class Protocol Problem

A mutual credit ledger can be perfectly designed and still fail if the cell is missing critical roles. In collapse conditions, missing a single essential capability (food handling, water sanitation, basic medical, fuel logistics) can collapse the whole network regardless of incentives.

So the protocol needs an explicit concept of a Minimum Viable Cell (MVC): a checklist + feasibility test that must pass before a cell is considered operational.

This is not bureaucracy. It is equivalent to the “valid chain” requirement in Bitcoin: without it, the system is not real.

---

95. Formal Definition: Minimum Viable Cell (MVC)

A cell C at time t is minimum-viable if:

1. Coverage Feasibility: the essential task coverage constraints have a feasible allocation:

   $$\exists x_{i,t}\ge 0:\quad \sum_t x_{i,t}\le s_i\ \forall i,\quad \sum_i a_{i,t}x_{i,t}\ge H_t^{\min}\ \forall t\in\mathcal{T}_{\text{essential}}$$

2. Energy Feasibility: energy constraints are feasible for the essential plan:

   $$\sum_{t\in\mathcal{T}} g_{t,j}(X_t)\le E_j + I_j\quad \forall j$$

3. Governance Feasibility: the cell has:

- a council G of size within bounds,
- quorum rules q,
- dispute process defined,
- emergency mode enabled.

4. Admission Feasibility: the cell has a mechanism limiting Sybil count S:

- sponsor bond and/or service bond,
- plus rate limits on privilege.

If any condition fails, the cell is in “formation” mode and should not promise survival-level autonomy.

---

96. Practical MVC: Role Buckets and Redundancy

The mathematical feasibility problem is exact but requires data you won’t have at onboarding. So we introduce a template that approximates feasibility with robust redundancy.

96.1 Essential Role Buckets

Define role buckets (with redundancy targets):

1. Food production & cooking
2. Water & sanitation
3. Heat/energy logistics (wood/fuel/solar maintenance)
4. Shelter repair & tool maintenance
5. Medical basics & caregiving
6. Childcare/dependent care
7. Procurement/transport
8. Coordination/admin/conflict resolution

96.2 Redundancy Rule

For each bucket t, require:

- at least r_t competent people
- not just hours.

Example redundancy targets for N\approx 80:

- Food: r=10
- Water/sanitation: r=4
- Energy: r=5
- Repairs: r=5
- Medical: r=3
- Care: r=6
- Transport: r=4
- Coordination: r=3

Why redundancy matters: if a single key person leaves or becomes ill, the cell must still satisfy H_t^{\min}.

Formally, you're trying to ensure feasibility under removal of a small subset of participants (a robustness constraint). Full robustness is NP-hard in general; redundancy is a practical surrogate.

---

97. MVC as a Robust Feasibility Condition (Formalized)

Let U be a set of potential shocks (illness, churn). For a robustness parameter k, require:

\forall S\subseteq M,\ |S|\le k,\quad \mathcal{F}(t;M\setminus S)\neq \varnothing

Meaning: after losing any k members, essential feasibility remains.

This is strong. In practice:

- set k=2 or 3 for key roles
- enforce redundancy thresholds r_t as a proxy.

---

98. Cell Formation Algorithm (Protocol-Level)

A new cell forms in stages, each with gates.

Stage 0: Formation

- members join as “candidates”
- ledger exists but with tight limits L_{\text{cand}} and no claim of survival autonomy

Stage 1: Role Declaration

Each candidate provides:

- hours available s_i
- role competencies a_{i,t} (self-declared + peer-confirmed)
- household structure h(i)
- energy assets (solar, wood access, tools)

Stage 2: Feasibility Check

The app runs a feasibility test:

- does the declared pool meet minimum redundancy targets?
- do total hours meet H^{\min}?
- do energy stocks/inflows meet minimum viable E for the plan?

If fails:

- app shows which role buckets are missing

- recommends recruiting or downsizing the cell

Stage 3: Governance Setup

- elect/appoint initial council
- define quorum, dispute rules
- enable emergency mode defaults

Stage 4: Activation

- set standard L
- enable escrow commitments for essentials
- allow essential bundle accounting to begin

This staged activation prevents early “false confidence” and reduces adoption blowback.

---

99. MVC Parameter Defaults for Wexford Pilot

For a Wexford-first pilot, I’d choose robustness over ambition.

Suggested initial targets:

- N = 40 to 60 for the first cell (tight governance, faster learning)
- form a “seed cell” with high competence density
- once stable, replicate to additional cells rather than scaling one cell too large

Why smaller first? Because your first real constraint is not math—it’s legitimacy and ops.

Example MVC thresholds for N=50

- Food: 6
- Water/sanitation: 3
- Energy/heat: 3
- Repairs: 3
- Medical: 2
- Care: 4
- Transport: 3
- Coordination: 2

And require at least:

- 35 adults with 15–25h/week each (or equivalent labor supply)
- clear access to at least one reliable local energy source (wood + basic storage)

---

100. “Survival Autonomy” Certification Levels

To avoid overpromising, define levels:

Level 0: Exchange-only

- casual swapping, no survival guarantee

Level 1: Essentials-lite

- food/community meals + minor repairs
- partial coverage

Level 2: Survival-capable

- full essential coverage constraints satisfied weekly
- energy budgeting active
- emergency mode tested

Level 3: Shock-resilient

- passes robustness threshold k
- federation links diversified
- proven recovery from stress scenario

The protocol can treat Level 2+ as requiring escrow commitments and tighter admission.

---

101. Big-Actor Intervention Angle: Why MVC Helps

MVC reduces intervention risk indirectly:

- fewer fraud incidents
- fewer crises and disputes
- less “it’s a scam currency” narrative fuel

A sloppy early rollout invites suppression; a disciplined, survival-focused rollout looks like mutual aid.

---

102. What Comes Next

At this point, the math/spec is strong. The next step is to tie it back to your original scenario papers and write a scenario appendix:

Addendum Part 6: Collapse Pathways (Systemic Looting → Trust Breakdown → Cell Formation Timeline in Ireland)

where we translate the macro thesis into phases and map which protocol features matter in each phase.

Say “go part 6” if you want that, and I’ll build it for the Ireland/Wexford context.

Addendum Part 6: Collapse Pathways and an Ireland/Wexford Cell Timeline

Systemic looting → trust breakdown → survival cells (mapped to your protocol features)

---

103. Why a “US-looking” credit thesis still matters in Ireland

Even if the mechanics (401(k), ERISA) are American, the pattern generalizes:

- Opaque private credit + valuation lag + “extend-and-pretend” delays visible defaults until refinancing becomes impossible.
- LMEs weaponize contracts to transfer value from passive capital to aggressive capital (“creditor-on-creditor violence”), and this “technology” is described as migrating to Europe by ~2025.
- Real-economy embedding: when leveraged structures sit inside healthcare, insurance, and infrastructure, the unwind stops being “Wall Street” and becomes “the hospital/utility

bill.”

For Ireland/Wexford, the transmission channels are typically:

- pension fund holdings (direct or through global mandates),
- bank/credit tightening and funding costs,
- insurer balance-sheet stress / delayed claims / repricing,
- energy/infrastructure pricing,
- unemployment and SME failure (especially in services and construction).

You don’t need “hyperinflation” to get collapse dynamics; you just need trust + liquidity + essential services to degrade.

---

104. A Phase Model for “Systemic Looting → Local Survival Cells”

I’ll structure this into 6 phases, each with:

(a) what is happening in the macro system, (b) what ordinary people feel, (c) what your protocol must already have turned on.

Phase 0: Quiet deterioration and narrative control

Macro: valuation lag, shadow defaults (e.g., bad PIK/distressed exchanges), and delayed loss recognition.

Local feel: “everything is fine” in headlines, but prices rise, services degrade, jobs get weird.

Protocol requirement: build quietly as mutual aid coordination, not “new money”.

Features that matter now

- Cellular architecture (no global target)
- No yield, no convertibility promise
- MVC formation gates (you only activate when actually viable)

Phase 1: Credit tightening and “selective rule-of-law”

Macro: refinancing wall accelerates extraction; LMEs concentrate pain on passive holders; documentation risk replaces credit risk.

Local feel: SMEs can’t roll debt; layoffs; “good borrowers” still get funding, others get cut off; anger rises.

Protocol requirement: provide partial substitution: people earn access to essentials via work even as cash economy frays.

Features that matter now

- Hard credit limits L (bounded loss)
- Admission friction (sponsor/service bonds)
- Escrow commitments for essentials (anti-flake when stress begins)

Phase 2: Real-economy embedding bites (healthcare/insurance/infrastructure)

Macro: the “octopus” feedback loop: price hikes to service floating-rate debt, breaking point, revenue collapse, credit freeze, contagion.

Local feel: insurance premiums jump, claims get harder/slower; healthcare access degrades; utility bills become political.

Protocol requirement: move from “exchange” to survival scheduling: guaranteeing minimal task coverage weekly.

Features that matter now

- Survival scheduler (coverage constraints)
- Households (dependents don’t get pushed to -L)
- Dispute process (legitimacy under stress)
- Emergency mode (Stressed triggers)

Phase 3: Exit-liquidity dynamics and public anger

Macro (pattern): when institutions are saturated and distributions dry up, the system hunts for new liquidity; in the US paper this is framed as moving private assets into defined-contribution flows via structural “adapters” (CITs, liquidity sleeves), with risks surfacing in crisis.

Local feel: “Why are ordinary people the bagholders?” trust in pensions, funds, and regulators collapses.

Protocol requirement: your system must not resemble an investable product. It must look like local accounting + mutual aid.

Features that matter now

- No investment framing
- No centralized treasury
- Open spec + local instances (gift-to-humanity model without a head)

Phase 4: Energy becomes the bottleneck

Macro: the infrastructure layer becomes strategic; “AI power crunch” narratives justify private control structures (“take-or-pay” style tollbooths) and competition for energy.

Local feel: heating/cooking/transport become unstable; rationing emerges formally or informally.

Protocol requirement: the system must enforce energy feasibility, not just “credits”.

Features that matter now

- Energy stock/flow accounting layer (physical, auditable)
- Energy-triggered emergency mode
- Rationing procedures (explicit humanitarian floors)

Phase 5: Institutional hostility / intervention pressure

Macro: as parallel coordination tools grow, states and large actors look for chokepoints: app stores, hosting, banking rails, identity providers, narrative control. (This follows from the “control surface” logic; and your SI premise amplifies it.)

Local feel: platforms disappear; accounts frozen; “illegal money” narratives; infiltration.

Protocol requirement: the system survives by not having chokepoints and by keeping governance human/local.

Features that matter now

- Offline-first operation
- Severability + federation caps \beta (damage containment)
- AI not in the control loop (advisory only)

---

105. Wexford-first: a concrete timeline that matches these phases

Step A: Build a Level-1 “Essentials-lite” cell (8–12 households)

Target: prove the social mechanics before claiming autonomy.

Deliverables:

- household accounts + basic ledger
- commitments for a single essential rota (e.g., community meals + transport)
- dispute procedure (boring, procedural, transparent)

Step B: Upgrade to Level-2 “Survival-capable” (40–60 people)

Target: pass MVC feasibility templates (roles + energy plan).

Deliverables:

- survival scheduler turned on (weekly coverage)
- energy inventory tracking for 1–2 carriers (wood + electricity storage)
- emergency mode rehearsals (tabletop exercises)

Step C: Replicate to 3–5 cells in the county

Target: replication beats scaling.

Deliverables:

- federation links with \beta \le 0.05
- severability drills (simulate a cell going offline)
- publish the “cell starter kit” as the gift-to-humanity artifact

---

106. Where your uploaded papers plug directly into the narrative of Part 6

These are the “load-bearing” scenario anchors:

- Silent, engineered credit disintegration + LMEs + extend-and-pretend until 2026 wall → why trust breaks suddenly when recognition becomes unavoidable.
- Shadow defaults / bad PIK / distressed exchanges and valuation failure → why official data lags lived reality.
- LME playbook + Europe migration → why “this can’t happen here” is a bad assumption.
- Embedded real-economy feedback loop + infrastructure/energy framing → why essentials and energy must be first-class in the protocol.
- Retail-liquidity pipeline logic (pattern of substituting new, less sophisticated capital) → why public anger rises and legitimacy collapses.

---

107. What comes next

If you want to shift from “paper spec” to “design spec,” the next most valuable thing is:

Part 7: Product Requirements Document (PRD) for the Wexford pilot, including:

- exact user roles/screens,
- onboarding and MVC checks,
- transaction + commitments UX,
- governance/dispute flows,
- offline-first architecture choices,
- and a test plan that simulates Phase 2–5 stressors.

Say “go part 7” and I’ll write that PRD in the same rigorous style (but now grounded in concrete app flows).

Addendum Part 7: PRD for the Wexford Pilot

Product Requirements Document for a Level-1 → Level-2 survival-capable cell

---

0. Purpose and Non-Goals

Purpose

Build a local-first mutual credit + survival scheduling app that can support a Wexford pilot cell, starting as Essentials-lite (Level 1) and graduating to Survival-capable (Level 2).

Non-goals (explicit)

- Not a token, not an investment product, not yield-bearing.
- Not a global network; replication is via multiple independent cells.
- Not a replacement currency; it is a coordination and accounting tool.

---

1. Scope

Level 1: Essentials-lite (Pilot)

Minimum viable to run:

- household accounts
- mutual credit ledger with hard limits
- basic task postings + commitments for 1–2 essential rotas
- basic dispute process (human)

- offline-tolerant operations (eventual sync)

Level 2: Survival-capable

Adds:

- MVC (Minimum Viable Cell) formation gates
- survival scheduler (coverage constraints)
- energy inventory layer (at least wood + electricity)
- emergency mode (Stressed/Panic)
- limited federation hooks (optional; default off)

---

2. Users and Roles

2.1 Roles

- Member: regular participant, belongs to a household.
- Household Admin: manages household membership, views household balance, accepts bundle debits.
- Task Lead: creates recurring tasks/rotas; confirms completion.
- Council Member: participates in governance actions (admissions, disputes, freezes).
- Auditor (rotating): checks energy stock records and rota integrity (procedural legitimacy role).

2.2 Identity Model

- Cell-scoped identity (public-key backed) + human-readable alias.
- Admission is local and council-approved.
- No global identity provider required.

---

3. Core Data Objects (MVP Schema)

3.1 Household

- household_id
- members[]
- balance b_h
- limit L_h
- bundle_profile (adult count, dependent count)
- status (active/frozen)

3.2 Member

- member_id
- household_id
- reputation R_i (advisory)
- roles/skills (self + peer-confirmed)
- availability s_i
- status (active/frozen/candidate)

3.3 Transaction (T1)

- tx_id
- payer_household
- payee_household
- value v
- timestamp
- description
- signatures: payer + payee (or payer + task lead for work-crediting)

3.4 Commitment (C1)

- commit_id
- household (or member, depending on mode)
- value v
- due_time
- task_id
- reserved r_h update
- status: created/fulfilled/cancelled/disputed

3.5 Task

- task_id
- category (Food/Water/Energy/Repairs/Care/etc.)

- hours
- schedule (one-off / recurring)
- lead_id
- required_skills
- participants[]
- status: open/assigned/done

3.6 Energy Stock

- carriers: wood, diesel, electricity (start with wood + electricity)
- E_j current stock
- I_j expected inflow this week
- audit_log[] (simple readings, photos optional, human signoff)

3.7 Dispute

- dispute_id
- parties
- claim type (non-delivery, quality, fraud, harassment, etc.)
- evidence (simple)
- status + outcomes

---

4. Invariants (Hard Requirements)

These must be enforceable at the validation layer.

Ledger invariants

- Conservation within a cell:

  $$\sum_h b_h = 0$$

- Hard floor:

  $$b_h \ge -L_h$$

- Escrow safety:

  $$b_h - r_h \ge -L_h,\quad r_h\ge 0$$

Federation invariants (Level 2+, optional)

- $\sum_k B_k=0$
- $|B_k|\le \beta \Lambda_k$

Energy feasibility invariants (Level 2)

- planned consumption cannot exceed availability:

  $$\sum_t g_{t,j}(X_t)\le E_j + I_j$$

---

5. Feature Requirements

5.1 Onboarding and Admission

User story: A new household wants to join.

Flow:

1. Create “candidate” identity + household.
2. Declare: adults/dependents, skills, weekly hours availability, energy assets/tools.
3. Council votes admission.
4. Household starts with:
    - tight limit L_{\text{new}}
    - service bond requirement (complete k hours before full privileges)

Acceptance criteria:

- no one enters with a high spend limit
- privilege increases are rate-limited

5.2 Household Accounting (mandatory)

User story: Dependents must receive essentials without being individually excluded.

Rules:

- essentials are debited to household, not individuals
- work credits accrue to household, credited via task completion

UI requirements:

- household balance dashboard

- upcoming debits (bundle forecast)
- reserved commitments display

5.3 Ledger and Transactions (mandatory)

User story: A member provides a service, earns credits; later spends credits.

Transaction modes:

1. Work credit (task lead confirms): household +v
2. Exchange (two households agree): payer −v, payee +v
3. Bundle debit (weekly): household −C_h, essential-provision households +C_h split

Validation:

- reject if payer would cross floor considering reserves

5.4 Commitments and Rotas (mandatory for Level 1)

User story: Food rota shifts must happen.

Rules:

- essentials commitments are escrowed (C1)
- creating a commitment reserves capacity (reduces spendable balance)
- fulfillment releases reserve + credits household

UI:

- calendar/rota view
- “my commitments” with warnings
- completion confirmation workflow

5.5 Disputes and Freezes (mandatory)

User story: A task wasn’t delivered; conflict must not collapse trust.

Rules:

- disputes are created with minimal evidence objects
- council can freeze an account
- outcomes: compensating transaction, reputation hit, limit reduction, expulsion
- any action must preserve invariants

UI:

- dispute timeline
- vote view for council
- transparent resolution log

5.6 MVC Gate (Level 2)

User story: The cell should not claim “survival capable” unless it is.

Inputs:

- declared skills and availability
- role redundancy targets
- energy stocks and inflows
- essential task minimums

Outputs:

- pass/fail per category
- “missing roles” checklist
- recommended recruitment targets

Acceptance criteria:

- app can objectively say: “Not viable yet.”

5.7 Survival Scheduler (Level 2)

User story: Ensure weekly essential coverage constraints are met.

Functionality:

- define H_t^{\min} per essential category
- assign tasks to meet coverage
- highlight deficits

- prioritize matching for households near floor to critical tasks (earn-back)

Constraint engine requirement:

- deterministic feasibility checks
- no AI required for decisions

5.8 Energy Layer (Level 2)

User story: Heating/cooking feasibility must be enforceable.

Features:

- energy stock entry + simple audit trail
- weekly energy budget dashboard
- link tasks to energy flows (wood gathering increases inflow; cooking consumes)
- compute stress index S_E

5.9 Emergency Mode (Level 2)

User story: Panic requires hard mode changes immediately and legitimately.

Triggers:

- floor mass F(t)
- disputes/churn
- energy stress S_E(t)

Actions (deterministic):

- Stressed: tighten new limits, expand escrow scope, reduce \beta
- Panic: freeze federation, reduce limits boundedly, freeze non-essentials, ration bundles

UI:

- banner showing current mode and why
- clearly logged changes (legitimacy)

---

6. Architecture Requirements

6.1 Local-first / Offline-tolerant

- App must function during connectivity loss.
- Local event log; sync later.

6.2 Deterministic Core

- Validation rules must be deterministic.
- AI (if used) is advisory only and cannot modify state.

6.3 Minimal Central Services (if any)

For pilot convenience you may host a sync service, but it must not be a single point of failure:

- multiple admins can run it
- cell can export/import logs
- ability to fork and continue

---

7. Security Requirements (Pilot Grade)

7.1 Admission friction

- sponsor/service bond required
- new households have low initial L

7.2 Privilege rate limiting

- L increases slowly; hard bounded
- council cannot “print”

7.3 Audit roles

- rotating human audit for energy and rota integrity

7.4 Tamper evidence

- event logs signed
- clear provenance of changes

---

8. Success Metrics for the Wexford Pilot

Level 1 metrics (4–8 weeks)

- rota completion rate > 90%
- dispute rate stable and resolving
- no household hits hard floor unexpectedly
- user retention > 70%

Level 2 metrics (8–16 weeks)

- MVC pass consistently (roles + energy)
- essential coverage constraints met weekly
- panic mode tested in drills and recovered
- bounded-loss property observed in simulated defection

---

9. Pilot Plan (Concrete)

Week 0–2: Formation

- recruit 8–12 households
- agree essential categories (start with meals + transport)
- run Level 1 ledger + commitments

Week 3–8: Stabilize

- tighten procedures, disputes
- add household bundle debits for 1 essential

Week 9–16: Level 2 upgrade

- define H_t^{\min} targets
- add energy inventory + S_E
- implement emergency mode
- run tabletop shock drills

---

10. What You Should Decide Next (One choice)

To proceed from PRD to build spec, the next decisive choice is:

Do you want the first pilot to focus on:

- (A) Food rota + shared meals (highest legitimacy, easiest to verify), or
- (B) Repair/maintenance + tools (lower perishability, good for skill matching), or
- (C) Energy/heat (highest strategic value, but politically sensitive and hardest to audit)?

Pick A/B/C and I'll write the next artifact: a detailed UX flow + screen list + state machine mapping for the pilot, fully consistent with the invariants.

Good — A (food rota + shared meals) is the right first pilot. It has the best mix of: high legitimacy, easy verification, immediate value, and low "currency" optics.

Below is the next artifact: Level-1 UX flow + screen list + state machine mapping for the Wexford pilot, focused on community meals as the first essential.

Addendum Part 8: Wexford Pilot UX and State Machine

Food rota + shared meals (Level 1 Essentials-lite)

---

1) Core Concept for the Pilot

What the cell actually does in Level 1

- Runs 2–4 shared meal events per week (e.g., Tue/Thu/Sun).
- Uses the app to coordinate:
    - who cooks
    - who procures
    - who transports
    - who cleans
    - who receives meals
- Uses household mutual credit to:
    - credit those doing the work
    - debit households receiving meals
- Uses escrowed commitments so shifts get done.

This is enough to prove:

- commitments
- legitimacy
- dispute handling
- household accounting
- bounded loss

without making big promises.

---

2) Minimal Roles for the Food Pilot

- Member (household participant)
- Meal Lead (responsible for one meal event)
- Council (small; admission + disputes only)
- Pantry Steward (tracks ingredients inventory; can be Meal Lead)

---

3) Screen List (MVP UX)

S0. Welcome / Cell Join

- Join existing cell (invite code / QR / in-person admission)
- Create local identity
- Create household (adults/dependents)

- Declare availability (hours/week)
- Declare skills tags: cook / driver / prep / cleanup / procurement

S1. Home Dashboard

- Next meal event countdown
- Household balance b_h and limit L_h
- “Your commitments” (with due times)
- “This week’s meal plan” summary
- Alerts: near floor, missed commitments, disputes

S2. Meal Calendar

- List of upcoming meals
- Each meal shows:
    - time/location
    - current role fill status (Cook/Procure/Transport/Cleanup/Serving)
    - servings planned vs households registered

S3. Meal Event Detail

For a selected meal:

- roles needed and hours per role
- “commit to role” button (escrow occurs)

- list of households registered to receive meals
- ingredient list + procurement tasks
- completion confirmation controls (Meal Lead)

S4. Commitments

- List of active commitments
- Escrowed value r_h shown
- “Fulfill” workflow (check-in + lead confirmation)
- “Request swap” workflow

S5. Transactions

- Ledger feed:
    - work credits earned
    - meal debits applied
    - transfers between households (optional)
- Each entry is readable and explainable (legitimacy)

S6. Household

- Members in household
- Bundle profile (adults/dependents)
- Weekly forecast: expected meal debits

- “Add/remove member” (requires council approval)

S7. Disputes

- Create dispute (missed shift / quality / harassment / fraud)
- Evidence minimal: timestamps, check-in, lead notes
- Council vote timeline + outcome log

S8. Council Panel (restricted)

- Candidate admissions
- Freeze/unfreeze
- Dispute resolution voting
- Parameter view (only within bounds)

---

4) State Machine: Food Meal Event Lifecycle

4.1 Meal Event States

Let meal event M have states:

1. Planned
2. Open for commitments

3. Locked
4. In progress
5. Completed
6. Settled
7. Disputed (overlay)

Transitions

Planned → Open

- Meal Lead created event with:
    - required roles R
    - hours per role
    - serving capacity S
    - location/time

Open → Locked

- occurs when either:
    - all essential roles filled, OR
    - lock deadline hit (e.g., 24 hours before meal)
- if roles not filled at deadline, Meal Lead can:
    - reduce servings (ration)

    - ask council to request volunteers

Locked → In progress

- at meal start time

In progress → Completed

- Meal Lead confirms meal produced and served

Completed → Settled

- ledger postings executed:
    - work credits to households who worked
    - meal debits to households who received

Any state → Disputed

- if a role unfulfilled, misconduct, non-delivery, etc.

---

5) Commitment Mechanics for Food Roles (Escrowed C1)

Each role commitment is a C1 reservation:

When a household commits to a role

- commitment value v = hours promised (typically 2–5)
- reserve increases:

  r_h \leftarrow r_h + v
- validation requires:

  b_h - r_h - v \ge -L_h

UX effect: you can't over-commit if you're already near the floor.

When commitment is fulfilled

- reserve decreases:

  r_h \leftarrow r_h - v
- household earns:

  b_h \leftarrow b_h + v

  (Work credit is just a transfer from a "meal production account" if you want strict accounting, or direct credit minted with matching debits at settlement; simplest is to credit workers and debit consumers at settlement to preserve conservation.)

If commitment is missed

- Meal Lead flags "missed"
- Dispute process opens automatically
- Council may impose:
    - reputation decrease

- temporary reduction in L_h
- requirement to complete makeup shift

No automatic punitive debits beyond the reserve logic; penalties must be legitimate.

---

6) Meal Debits: How Households Pay for Meals

For Level 1, keep it extremely simple:

Meal debit rule

Each household registered to receive meals pays a fixed debit per meal:

- Adults: c_A credits
- Dependents: c_D credits (smaller)

So for household h attending meal M:

$$\text{Debit}(h,M) = c_A \cdot n_h + c_D \cdot d_h$$

Conservation at settlement

Let total meal debits be:

$$D_M = \sum_{h \in \text{attendees}} \text{Debit}(h,M)$$

Let total work credits issued for meal roles be:

$$W_M = \sum_{\text{roles}} v_{\text{role}}$$

To preserve zero-sum, you set:

- workers receive W_M
- attendees pay D_M

  and the difference goes to (or is paid from) a Pantry Account P (a cell-owned internal account) that tracks stockpiling:

b_P \leftarrow b_P + (D_M - W_M)

This pantry account is not “money printing”—it’s an internal balance that represents stored labor claims for future ingredient procurement and prep work.

Hard constraints still apply to pantry and it remains visible.

---

7) Parameter Defaults for the Food Pilot (Wexford)

Start conservative:

- Unit: 1 credit = 1 hour
- New household limit: L_{\text{new}} = 8
- Standard household limit after bond: L_{\text{adult}} = 20
- Dependent contribution to limit: L_{\text{dep}} = 6, capped at 2 dependents for limit calc
- Meal debit per adult per meal: c_A = 1.5
- Meal debit per dependent per meal: c_D = 0.8
- Typical role commitments:
  - Cook: 4h
  - Procurement: 3h

    - Transport: 2h
    - Cleanup: 2h
    - Serving: 2h

This makes a meal event roughly balance:

- 10–14 hours of work covers 8–10 households attending.

We'll tune once we see participation.

---

8) Acceptance Criteria for the Food Pilot (What "works")

In the first 4–8 weeks, success is:

1. Meals happen reliably
    - role fill rate > 90%
2. Ledger stays socially legitimate
    - households understand why balances move
3. No runaway floor mass
    - few households stuck at -L
4. Disputes don't destroy trust
    - disputes resolved with clear procedure

5. App is not perceived as “money”
    - it is perceived as “meal coordination + fair work tracking”

---

9) Next Deliverable

If you want to move from PRD to build-spec, the next thing I should write is:

Data model + API contracts + deterministic validation module for:

- household ledger updates
- commitments lifecycle
- meal settlement batch posting
- dispute state transitions

Say “go build spec” and I’ll produce that as the next part.

Understood — staying at PRD level. I’ll extend the PRD to include Track B: Repair/Maintenance + Tools, still consistent with the same invariants and the Level-1 → Level-2 path.

PRD Extension: Track B — Repair/Maintenance + Tools

Lower perishability, high skill-matching, strong “real economy” legitimacy

---

1) Why Track B Is a Strong Second Vertical

Repair/maintenance is ideal because:

- outputs are durable (a fixed roof matters for months)
- verification is straightforward (the thing is repaired or it isn't)
- it builds "infrastructure competence" inside the cell (critical in collapse)
- it grows tool inventory (a real asset base)
- it reduces intervention optics (looks like a repair co-op, not money)

In your combined scenario (systemic looting + energy dominance), repair is also the pathway to:

- insulating homes,
- maintaining solar/battery systems,
- tool-making and reuse,
- reducing dependence on external supply chains.

---

2) Product Goal for Track B (Level 1)

Provide a reliable way for households to:

- request repairs,
- schedule work parties,
- borrow tools,
- track tool custody and maintenance,
- credit labor fairly,
- and resolve disputes procedurally.

Still not a marketplace. It's a maintenance scheduler + tool library + mutual credit.

---

3) Core User Stories

US-B1: Request a repair

"As a household, I need help fixing something (roof leak, plumbing, wood stove, gate)."

US-B2: Offer repair labor

"As a skilled member, I want to contribute hours and earn credits."

US-B3: Join a work party

"As a general helper, I can do support labor (carry, clean, prep, fetch)."

US-B4: Borrow a tool

"I need a drill/ladder/strimmer; I'll borrow it and return it."

US-B5: Maintain shared tools

"Someone must maintain/charge/sharpen tools and track wear."

---

4) Roles (Track B)

- Requester (household needing repair)
- Repair Lead (responsible for diagnosis, plan, safety)
- Helper (general labor)
- Tool Steward (library inventory and custody)
- Council (disputes, freezes, admissions)

---

5) New Data Objects (Track B)

5.1 Repair Job

- job_id
- requesting_household
- category (plumbing, electrical, carpentry, heating, general)
- severity (non-urgent / urgent / safety-critical)
- description
- location (approx / cell-zone)
- required_skills[]
- required_tools[]
- estimated_hours (lead estimate)
- status: requested → triaged → scheduled → in_progress → done → settled → disputed
- lead_id
- helpers[]

5.2 Tool Item

- tool_id
- type (drill, saw, ladder, multimeter)
- owner (cell library or household-contributed)

- condition (A/B/C + notes)
- custody_household (current holder)
- deposit_required (optional, in credits)
- maintenance_interval (hours of use)
- status: available / checked_out / overdue / maintenance / retired

5.3 Tool Checkout (Chain of Custody)

- checkout_id
- tool_id
- from (tool library or prior household)
- to (household)
- start_time, due_time
- condition_out, condition_in
- signatures (steward + borrower)

5.4 Tool Deposit Reserve (Optional, but recommended)

- uses escrow reserve r_h to create a deposit hold:
    - borrow requires reserving d credits
    - returning tool releases reserve
    - damage can trigger dispute → council can convert some reserve into compensation transfers

This creates strong incentives without needing courts.

---

6) Screen List (Track B UX)

B1. Repairs Dashboard

- “Open requests”
- “Scheduled jobs”
- “Urgent jobs”
- “Jobs needing a lead”

B2. Create Repair Request

- photo optional
- description + urgency
- availability windows
- consent checkbox for home access (procedural legitimacy)

B3. Repair Job Detail

- job plan + required roles
- “claim lead” (requires minimum reputation/approval)
- “commit to helper shift” (escrowed)
- required tools list + borrow buttons

- completion confirmation

B4. Tool Library

- searchable inventory
- status + condition
- “request checkout”
- “report damage” / “request maintenance”

B5. Tool Checkout Flow

- choose duration
- reserve deposit (escrow)
- steward approves
- return flow with condition check

B6. My Work

- upcoming repair commitments
- hours logged
- jobs completed
- disputes

(Plus shared screens from Track A: Home, Household, Ledger, Commitments, Disputes, Council.)

---

7) State Machines

7.1 Repair Job Lifecycle

Requested → Triaged

- Repair Lead or council triage:
    - classify severity
    - define required skills/tools
    - estimate hours

Triaged → Scheduled

- pick date/time
- fill roles (Lead + helpers)
- escrow commitments created

Scheduled → In progress

- start time reached; check-in confirms presence

In progress → Done

- Lead marks repair complete; requester household confirms “accept” (simple signoff)

Done → Settled

- ledger postings:
    - credit workers for hours logged
    - debit requester household (or debit partly + cell subsidy if essential/safety)
    - tool deposits released if returned

Any → Disputed

- missed work, poor quality, damage, safety incident, tool loss

7.2 Tool Checkout Lifecycle

Available → Checked out

- borrower escrows deposit d
- custody changes to borrower household

Checked out → Overdue

- due time passed; warnings
- repeated overdue triggers council action

Checked out → Returned

- steward inspects; release escrow

- if damaged: open dispute

Returned → Maintenance

- if condition requires maintenance; create maintenance task

---

8) Ledger Rules for Repairs (Simple, Legitimate)

8.1 Work credits

Repair work is credited in labor-hours:

- Lead logs hours for each participant
- requester confirms completion (or dispute)

Credits:

b_{h_i} \leftarrow b_{h_i} + \text{hours}_{i}

8.2 Requester debits

Requester household pays for the repair in credits:

b_{h_{\text{req}}} \leftarrow b_{h_{\text{req}}} - V

where V is total logged hours (or a capped estimate if you want predictability).

8.3 Pantry/Tool Library Account

If you want a shared tool library to grow, you need a visible account b_T:

- small per-job overhead (e.g., 5–10%) can flow to tool maintenance:

  b_T \leftarrow b_T + \tau V

  This must be transparent and locally approved (to avoid “tax” resentment).

Important: b_T is still inside conservation—every credit added is matched by debits elsewhere.

---

9) Safety and Legitimacy Requirements (Repairs Are Higher Risk Than Meals)

Repairs introduce:

- property damage risk
- safety risk (electricity, ladders)
- accusation risk

So PRD must include:

- explicit consent steps for access
- safety checklists for certain job categories
- incident reporting flow
- council escalation thresholds

Keep it boring and procedural.

---

10) Level 1 → Level 2 Upgrade Path for Track B

Level 1 (Repair-coop)

- repair scheduling
- tool library + custody + deposits
- hours-to-credits settlement
- disputes

Level 2 (Infrastructure survival)

Add:

- essential maintenance coverage (minimum weekly hours):
    - heating systems
    - water pumps/filters
    - insulation/winterization
- energy coupling (repairs that increase energy inflows, e.g., solar maintenance)
- emergency mode: prioritize safety-critical repairs

This is where Track B becomes the backbone of the energy-constrained model.

---

11) Metrics for Track B Success

Operational

- % repair requests triaged within 48h
- % scheduled jobs completed on time
- mean time to repair for urgent jobs

Trust

- dispute rate per job
- tool loss/damage rate
- overdue rate

Economic

- distribution of balances (avoid many households pinned at -L)
- throughput (hours/week completed)

Resilience

- ability to complete safety-critical repairs even during stress mode

---

12) Next PRD Step (No Build Yet)

To continue the PRD rigorously, the next section should define policy defaults for Track B:

- which repairs are “essential” vs optional
- whether requester always pays full cost or cell subsidizes safety-critical jobs
- what tool deposit values should be (and how bounded-loss interacts with deposits)
- minimum reputation/approval required to be a Repair Lead

If you say “go”, I’ll write those policy defaults as a clean PRD subsection with parameter ranges and the reasoning tied back to the bounded-loss and legitimacy constraints.

PRD Subsection: Track B Policy Defaults (Repair/Maintenance + Tools)

Essential classification, subsidies, deposits, and permissions (with bounded-loss + legitimacy constraints)

B-Policy-1: Repair Classification System

Repairs need a triage policy that is:

- simple enough to be legitimate,
- strict enough to prevent exploitation,
- and aligned with survival constraints.

Classes

Class 0 — Cosmetic / Convenience

- e.g., painting, aesthetics, non-urgent upgrades

Class 1 — Standard Maintenance

- e.g., stuck door, minor plumbing, small patch job, appliance troubleshooting

Class 2 — Safety / Habitability Critical

- e.g., roof leak in winter, electrical hazards, heating failure, sanitation failures, broken windows/doors affecting security and heat

Class 3 — Infrastructure Critical (Level 2+)

- e.g., water systems, solar/battery maintenance, generator repair, community kitchen equipment, communal storage

Policy consequence

- Class 2–3 are eligible for priority scheduling and cell subsidy.
- Class 0–1 are paid mostly by requester.

This stops the system being drained by “nice-to-have” jobs when survival capacity is scarce.

B-Policy-2: Who Pays for Repairs? (Requester vs Cell Subsidy)

If requester always pays full cost, the poor get locked out of safety repairs. If the cell always subsidizes, you invite free riding.

So we define a two-part rule:

Rule B2.1: Standard Jobs (Class 0–1)

Requester pays full value:

$$\text{Debit}_{\text{req}} = V$$

where V is logged labor hours (or capped estimate; see below).

Rule B2.2: Safety/Habitability Jobs (Class 2)

Requester pays a fraction, cell covers remainder:

$$\text{Debit}_{\text{req}} = \gamma V,\quad \text{Debit}_{\text{cell}} = (1-\gamma)V$$

with:

$$\gamma \in [0.3, 0.7]$$

Default:

$$\gamma = 0.5$$

Cell coverage is debited to a Cell Maintenance Account b_M (an internal account funded by small overhead from other transactions or explicit contributions). This keeps conservation intact and keeps subsidies transparent.

Rule B2.3: Infrastructure Critical (Class 3, Level 2+)

- treated as essential coverage tasks
- scheduled like survival constraints
- funded by the cell as a whole, with household debits handled via the essential bundle mechanism (not per-job “billing”)

This prevents fights over “who pays” when the job is literally infrastructure survival.

B-Policy-3: Pricing Repairs (Avoiding Market Pathologies)

You have two legitimate options. Pick one for the pilot.

Option 1: Pure Labor-Hour Pricing (recommended early)

- V = total logged hours
- no haggling
- minimal conflict

Pros: simple, fair, discourages “price gouging” narratives

Cons: doesn’t reflect scarce expertise

Option 2: Labor + Scarcity Multiplier (later, if needed)

If some skills are truly scarce (e.g., electrical), you can apply a bounded multiplier:

V = \sum_i \alpha_{s(i)} \cdot h_i

where:

- h_i is hours logged by worker i
- \alpha_{s}\in[1, \alpha_{\max}] depends on skill category
- \alpha_{\max} should be small (e.g. 1.5 or 2.0)

This preserves legitimacy while allowing scarce expertise to be rewarded.

Pilot default: Option 1 only.

B-Policy-4: Estimation and Cost Predictability (Anti-Dispute Design)

Disputes explode when a requester expects “2 hours” and gets billed “10.”

So every job has a lead estimate E and a cap policy.

Default cap policy

- If final hours V \le 1.25E: requester pays V
- If V > 1.25E: excess requires requester re-approval mid-job, otherwise excess is paid by the cell maintenance account or treated as disputed.

This keeps the system procedurally legitimate.

B-Policy-5: Tool Deposits (Escrow) That Don’t Break Bounded-Loss

Tool custody is a prime attack surface:

- tools are real assets and scarce
- losing them harms the whole cell

But deposits must not become:

- punitive money,
- or a way to force people below the hard floor.

Deposit mechanism

Borrowing tool u requires reserving d(u) credits:

r_h \leftarrow r_h + d(u)

with feasibility:

b_h - r_h - d(u) \ge -L_h

Meaning: if you're near the floor, you can't borrow expensive tools without earning capacity first. This is correct.

Deposit sizing policy

Set deposits to approximate:

- replacement difficulty (not fiat price)
- scarcity value in labor-hours

Defaults:

- small tools (hand tools): d=1–3
- drill/strimmer: d=4–8
- ladder: d=6–10
- multimeter: d=3–6

Hard rule: deposits must be \le 0.5L_h for most households, otherwise you lock out everyone except the credit-rich.

Damage/loss outcome

If lost or damaged:

- automatic dispute
- council decides whether some portion of reserve converts into compensation transfers to Tool Library Account b_T

This preserves due process; no automatic "fines."

B-Policy-6: Permissions to Be a Repair Lead (Prevent Safety Failures)

Repairs can kill people (electricity, ladders). You need a permission system that is legitimate and robust against manipulation.

Default Lead Qualification Levels

- Lead-0 (General): doors, basic carpentry, non-critical tasks
- Lead-1 (Plumbing/Heating basic): non-gas plumbing, stove maintenance under checklists
- Lead-2 (Electrical / high risk): requires explicit council approval + peer endorsements

Rule

To claim Lead role in category c, member must satisfy:

- minimum participation history (e.g. 3 completed helper shifts)
- peer-confirmed skill tag
- no active disputes
- council override possible for emergency

This is the anti-infiltration design: privilege grows slowly and locally.

B-Policy-7: Prioritization Rules (When Demand Exceeds Capacity)

When repairs backlog forms, the app must have a transparent ordering rule:

1. Class 3 (infrastructure critical)
2. Class 2 (safety/habitability)
3. Class 1 (maintenance)
4. Class 0 (cosmetic)

Tie-breakers:

- households closest to the floor get priority for subsidized essential fixes (to avoid spiral)
- but only for Class 2–3 (otherwise it's exploitable)

This ties directly to survival stability: you prevent "debt traps" from becoming homelessness and failure cascades.

B-Policy-8: Coupling Track B to Track A (Food Pilot)

This is important: repairs can support the food system.

Examples:

- community kitchen repairs
- refrigeration

- cooking equipment
- insulation/heat for cooking spaces

Policy: any repair that supports the food rota can be tagged “Food-critical” and treated as Class 2–3 for priority and subsidy.

That yields coherent “survival-first” behavior from two tracks.

B-Policy-9: Parameter Defaults Summary (Pilot)

- Pricing: labor-hours only (no scarcity multiplier yet)
- Subsidy fraction for Class 2: \gamma=0.5
- Estimate cap threshold: 25% over estimate requires re-approval
- Deposits: 1–10 credits, capped by household ability to reserve
- Lead permissions: slow, peer-confirmed, council can approve high-risk categories

---

If you want me to continue the PRD, the next piece that will make implementation inevitable (still without coding) is a full end-to-end flow narrative for 3 representative repair cases:

1. a trivial maintenance job (Class 1)
2. a safety-critical roof leak (Class 2)
3. a tool loss dispute with escrow + council outcome

Say “go cases” and I’ll write those flows as PRD-level “happy path + failure path” sequences.