NetWrap: A Deficiency-Resilient Protocol for Multilateral Netting with Preserved Bilateral Exposures

Daniel Aronoff

Draft date: April 29, 2026

Abstract

Multilateral netting reduces settlement balances, inventories, and balance-sheet usage in Treasury cash, repo, and other OTC markets, but existing methods impose a trade-off. Compression and central clearing lower gross obligations by rewiring exposures or concentrating them in a CCP, while atomic-cycle methods preserve bilateral exposures only by failing when any participant cannot fund. NetWrap is a wrapper that can sit atop any netting core for either a single asset or delivery-versus-payment trades. Parties with net outflows must first lock the required balances. If one cannot fund in full, NetWrap removes only the deficiency-linked portions of that party's sending assignments (with or without its corresponding receiving assignments), reinstates them bilaterally between the original counterparties, and re-nets the remainder. The mechanism executes nettable trades in finite iterations, preserves original counterparty risk for unfunded portions, extracts all available funded netting, and produces an auditable transcript for institutional or blockchain implementation.

Keywords: multilateral netting; counterparty risk; market design; settlement; cryptography; blockchain.
JEL: G10, G20, D47, D85, C61.
ACM CCS: Applied computing → Electronic commerce; Security and privacy → Cryptography; Computer systems organization → Distributed architectures; Applied computing → Operations research.
Author disclosure: The author has filed patent applications related to NetWrap. This paper presents the protocol and its properties as a research contribution independent of those filings.

Contents

1 Introduction

Netting is widely used in wholesale money and securities markets. By offsetting directed obligations, it reduces settlement volume, inventory requirements, and balance-sheet usage. The choice of netting mechanism also affects the exposure map: who remains exposed to whose default. Trade compression cancels offsetting contracts by arranging them into chains and cycles and replacing them with fewer contracts, often creating bilateral relationships between nodes that never directly traded. Central clearing novates bilateral directed obligations into contracts with a central clearing counterparty ("CCP"), enabling multilateral netting while concentrating counterparty risk in the CCP. Blockchain-based atomic-swap proposals can preserve bilateral exposures by netting along identified cycles, but they generally do so through all-or-nothing execution: if any required participant cannot fund or authorize its leg, the cycle does not settle.

In financial markets, counterparty identities determine credit limits, collateral terms, margin requirements, regulatory capital, close-out rights, and monitoring incentives. Trade compression can reduce gross obligations, but compression on chains and cycles can replace the original exposure map with new bilateral links among parties that never directly traded. Central clearing avoids arbitrary new bilateral links by interposing a CCP, but it concentrates counterparty risk in the CCP and requires margin, default-fund, access, and fee arrangements that may be costly when balance sheet and collateral are scarce. These frictions help explain why markets, such as OTC derivatives, bond and repo markets do not net all eligible transactions. The goal of NetWrap is to relax this tradeoff: it does so by clearing funded multilateral netting while preserving the original bilateral exposure map for any unfunded portions.

NetWrap is a protocol wrapper that can augment a chosen netting core without changing the arithmetic of the core itself. It adds a two-loop commit–verify–settle architecture that converts one-shot netting into a deterministic multi-round protocol robust to funding shortfalls. In the outer loop, nodes with net outflows must commit the required traded objects to a verifiable locked state before settlement. If a node cannot fund in full, an inner loop removes only deficiency-linked portions of that node's sending assignments, reinstates those portions bilaterally between the original counterparties, and re-nets the residual set. Iteration continues until all remaining netted obligations are funded. The resulting settlement extracts available funded netting, preserves the original exposure relationships for unfunded portions, and produces an auditable transcript suitable for institutional or blockchain implementation.

Viewed as a distributed-systems problem, the key technical challenge is partial commit fail-

ure during a multi-node completion phase. When one or more participants miss a lock deadline, naive implementations either abort and restart the entire coordination attempt, retry without a bounded-progress guarantee, or rely on operator discretion and non-verifiable logs. NetWrap specifies a restart-safe coordination layer: commit and completion instructions are bound to a protocol instance identifier (run_id), replacement-contract generations and cancellations are versioned, and each state transition is recorded in a tamper-evident transcript sufficient for deterministic replay and independent audit.

2 Background

In decentralized markets, agents (whom we also call "nodes") often enter numerous bilateral trades, such as delivery-versus-payment contracts, loans, or payment obligations. These trades create a web of interdependent directed obligations. Broker-dealers carry inventories to provide liquidity to the markets they intermediate, and required inventory generally increases with trading volume. In U.S. repo markets, broker-dealers are required to record as assets some cash flows that they do not retain but pass through in matched-book trades. The resulting increase in recorded assets interacts with bank regulations such as the Supplementary Leverage Ratio and can limit the volume of lending that bank-affiliated broker-dealers can provide [Aronoff et al., 2025]. These markets include some of the largest financial markets in the world, including the U.S. Treasury repo market, the U.S. Treasury secondary market, and global OTC interest-rate, foreign-exchange, and credit-derivatives markets. Netting techniques reduce settlement flows, inventory requirements, and balance-sheet usage by consolidating multiple directed obligations. Instead of settling each contract individually, nodes offset mutual obligations so that only net transfers are made. The challenge is that netting can also affect how credit risk—the risk of counterparty default—is distributed and preserved. The two methods most often used to achieve multilateral netting are trade compression and central clearing. Recent blockchain-native methods also identify cycles of directed obligations and execute atomic swaps along those cycles.

2.1 Trade compression

Trade compression is used in many over-the-counter (OTC) derivatives markets to tear up redundant initial directed obligations.[1] Trade compression allows multiple counterparties to cancel offsetting directed obligations collectively, thereby reducing gross notional amounts. In multilateral compression chains and cycles, nodes report their directed obligations to a trusted node that identifies and removes offsetting positions. Compression can, however,

[1] TriOptima's triReduce service is used for trade compression in global OTC interest-rate, foreign-exchange, and credit-derivatives markets [MarketsWiki, 2025, CLS Group, 2015].

create new counterparty exposures. For a chain or cycle of offsetting exposures, compression may obligate nodes at the ends of a chain, or non-adjacent nodes in a cycle, to become counterparties even though they did not initially contract with each other. Similarly, in a chain or cycle involving two traded objects, such as cash and Treasuries, a failure by a node that must fund the cash leg can cause default of the entire contract and produce intertwined residual exposures among the other nodes. This transfer and increased complexity of default risk and resolution can limit the use of trade compression. TriOptima's triReduce uses an optimization process to identify offsetting positions and cancel them in bulk. It relies on a trusted party to generate netting possibilities and new bilateral contracts between previously unrelated nodes, thereby altering the counterparty-exposure graph.

As an example of trade compression, consider directed obligations in units of a single object, say money, among nodes A, B, C, D, and E. A contract is represented by the tuple {node 1, node 2; object 1, object 2}, where node 1 sends object 1 to node 2 and node 2 sends object 2 to node 1; "-" indicates the absence of a second object. The contracts are $\{A, B; 2, -\}$, $\{B, C; 2, -\}$, $\{B, D; 1, -\}$, $\{D, C; 1, -\}$, $\{C, E; 1, -\}$, and $\{C, A; 1, -\}$. Figure 1 displays the flow of traded objects. One configuration is a chain $B[1] \to D[1] \to C[1] \to E$ and a cycle $A[2] \to B[2] \to C[1] \to A$, where bracketed numbers denote units sent. Trade compression would net flows on the chain and cycle, altering counterparty risk. After netting on the chain, B becomes obligated to send one unit to E, with whom it did not initially contract, exposing E to default by B. After netting on the cycle, A becomes obligated to send one unit to C, with whom it did not initially contract, exposing C to default by A.

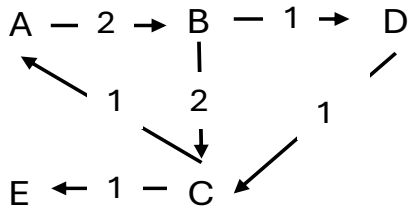

Figure 1: Initial directed obligations

2.2 Central clearing

In central clearing, a third party—the central clearing counterparty ("CCP")—novates the initial trades between nodes and interposes itself between counterparties, becoming the buyer to every seller and the seller to every buyer. This transforms a network of bilateral exposures into a hub-and-spoke model in which each agent is exposed to the CCP and the

CCP is exposed to all agents. In Figure 1, the initial directed obligations are novated and replaced by contracts between each counterparty and the CCP. For example, the contract $\{A, B; 2, -\}$ becomes $\{A, CCP; 2, -\}$ and $\{CCP, B; 2, -\}$. Each node then nets its directed obligations with the CCP. In the model of initial trades, central clearing results in four contracts: $\{B, CCP; 1, -\}$, $\{CCP, E; 1, -\}$, $\{A, CCP; 1, -\}$, and $\{CCP, C; 1, -\}$. The net inflow and outflow of object y for each node is the same as in trade compression. The difference is that each agent now has the CCP as its counterparty, which alters counterparty risk relative to the initial directed obligations.

2.3 Cycle detection

Several recent blockchain-based protocols detect cycles of directed obligations, net obligations on the cycle, and then execute an atomic swap in which node accounts are simultaneously debited and credited. These proposals differ in whether they use privacy-preserving multiparty computation to identify cycles and in how they execute the resulting atomic transfers.

Under these methods, the system first identifies cycles of directed obligations, e.g., $A[2] \rightarrow B[1] \rightarrow D[1] \rightarrow C[1] \rightarrow A$, from the initial directed obligations in Figure 1. The nodes self-report inventories. The protocol determines for each node whether its inventory plus incoming directed obligations are sufficient to meet outgoing directed obligations, and then constructs netting and settlement transfers for that cycle. Those transfers are executed as a single atomic all-or-nothing compound transaction: either every leg succeeds or none does, so the ledger never reflects a partially executed cycle.

The protocol proceeds only if all required tokens or other items for the cycle are committed and all signatures are collected. If any node lacks sufficient inventory or fails to authorize its leg, the compound transaction is not executed and the nodes remain in their pre-netting bilateral positions. Thus, a single failure prevents execution of the entire cycle and no new counterparty exposures are created by a failed attempt. For example, if A self-reports that it has one object in inventory but does not possess that object at the scheduled execution time, the cycle is not executed, and D and C do not net their directed obligations. Some specifications contemplate partial settlement within a cycle by splitting quantities and executing a sequence of transfers, but each step depends on the accuracy of node inventory disclosures. If the objects are not sent, or if the nodes do not possess the objects, the sequence of trades is canceled. These methods are therefore effectively one-shot per cycle: an all-or-nothing execution model rather than a multi-round re-netting protocol. Moreover, these methods are not designed to achieve full multilateral netting in all circumstances. For example, Project Ubin generates netting maps based on reported availability of liquidity.

If a node is short of objects, the trade is delayed to a later netting round. There is no guarantee of execution within a finite number of netting rounds.

Centralized liquidity-saving optimization Centralized settlement platforms also implement liquidity-saving optimization that is structurally similar (at a high level) to "feasible-subset" cycle methods: they search over queued directed obligations to identify a set of instructions that can settle subject to balance constraints. For example, CHIPS-style net settlement maintains prefunded node balances (and may require supplemental prefunding) and applies an optimization/search procedure over queued payment orders to select a settleable subset given available liquidity.[2] Likewise, T2S night-time processing supports optimization features such as technical netting, prioritization, and partial settlement to increase throughput under resource constraints.[3] These systems generally rely on authoritative balance/position records within the platform and they "recycle" unsettled instructions as resources change. Project Ubin Phase 2 is similar, but it relies on self-reporting of inventories rather than escrow-based re-netting or graph-preserving settlement [Monetary Authority of Singapore and Association of Banks in Singapore, 2017].

2.4 Netting cores, exposure maps, and execution failure

The examples in Figure 1 can be used to separate two distinct objects of analysis: the arithmetic of netting and the allocation of counterparty exposure. Interpret the flows first as a one-object network in which each directed edge is a transfer of money M. The figure contains the chain

$$B[1] \to D[1] \to C[1] \to E \text{ and the cycle } A[2] \to B[2] \to C[1] \to A,$$

where bracketed numbers denote units of M. A compression core can replace the chain with the direct obligation $B[1] \to E$. It can also compress the cycle into the single residual obligation $A[1] \to C$. These replacements preserve the terminal net positions in M, but they do not preserve the exposure map: E is now exposed to B, and C is now exposed to A, even though those pairs did not directly trade in the initial contract set.

Central clearing applied to the same Figure 1 flows produces the same net positions in M, but through a different exposure transformation. The CCP becomes the hub: B pays the CCP and the CCP pays E for the chain component, while A pays the CCP and the CCP pays C for the cycle component. The direct $B \to E$ and $A \to C$ exposures created

[2] See, e.g., public CHIPS netting/settlement descriptions and related technical materials.

[3] See, e.g., ECB/T2S documentation describing night-time settlement optimization, prioritization, and partial settlement.

by compression are avoided, but bilateral exposure to initial counterparties is replaced by exposure to the CCP. Thus, compression and central clearing may be equivalent as netting arithmetic while being different as financial contracts.

The same distinction applies when the relevant contracts involve two traded objects, such as delivery-versus-payment exchange of money M and Treasuries T. A two-object DvP trade overlays an M-flow and a T-flow and requires both objects to be delivered according to the settlement rule. Object-by-object netting can reduce the gross movement of both M and T, but the economic issue remains the same. A method that changes the terminal graph changes who bears loss if a party does not deliver the required M or T.

Cycle-based atomic methods and centralized liquidity-saving methods occupy a third point in the design space. They generally preserve the initial exposure map by executing only feasible existing obligations, or by delaying and recycling infeasible instructions. This avoids new bilateral exposures and avoids CCP substitution. The limitation is incompleteness under shortfalls. If one node in a cycle does not commit its required objects, the entire netting sequence is unwound and otherwise nettable trades are not netted (or are delayed until a later round).

Incumbent netting methods therefore exhibit three canonical failure modes. Compression achieves netting by rewiring exposures. Central clearing achieves netting by substituting CCP exposure for bilateral exposure. Atomic-cycle and liquidity-saving methods preserve bilateral exposure, but generally do so by aborting, delaying, or recycling obligations when a required leg cannot be funded. NetWrap is designed to separate netting arithmetic from these exposure-allocation choices: it can wrap a chosen netting core, require verifiable pre-funding of net outflows, remove deficiency-linked portions when funding fails, reinstate those portions bilaterally between the original counterparties, and re-net the funded residual set.

2.5 Financial and economic shortcomings of incumbent netting methods

The shortcomings of incumbent methods apply when counterparty identity is economically valuable. In Figure 1 chain, compression produces the economically simple settlement instruction $B[1] \to E$, but E did not initially choose B as a counterparty. In the cycle, compression produces $A[1] \to C$, but C did not initially choose A. In OTC derivatives markets this is a first-order constraint. Counterparty identity determines credit limits, ISDA netting sets, collateral support annexes, close-out rights, margin schedules, CVA and FVA calculations, regulatory-capital treatment, and internal risk limits. Firms may therefore impose compression tolerances that preserve selected bilateral relationships even when more aggressive compression would reduce gross notional further. Rewiring the exposure map is a deterrent to full utilization of trade compression [D'Errico and Roukny, 2021].

Treasury and repo markets raise a related but distinct issue. Central clearing avoids the arbitrary $B \to E$ and $A \to C$ links in Figure 1 by interposing a CCP. This can improve multilateral netting, reduce gross settlement flows, and relax some balance-sheet constraints in dealer intermediation [Duffie, 2017, Cochran et al., 2023, Bowman, Huh, and Infante, 2024]. But central clearing is not a costless way to preserve the economics of bilateral trading. It replaces bilateral exposure with CCP exposure, requires margin and default-fund resources, imposes clearing and sponsorship costs, and can require substantial legal-documentation, operational, collateral, and systems changes. These costs are particularly salient in low-margin, high-volume markets such as Treasury cash and repo.

Recent market assessments reinforce this point. BNY argues that Treasury central clearing will likely alter the economics of cash and repo transactions: although netting may reduce some balance-sheet costs, additional margin and sponsorship costs are expected to widen bid–ask spreads and increase the cost of repo funding and leverage [Wuerffel, 2025]. The same report notes that FICC estimated approximately $27 billion of additional margin from expanded clearing of indirect participants, with the possibility of a higher figure because the estimate was based on a subset of market participants [Wuerffel, 2025]. A 2025 survey similarly reports that 38 percent of firms expect U.S. Treasury central clearing to increase margin requirements by more than 25 percent, while 55 percent expect higher regulatory capital costs [DTCC et al., 2025]. The survey also identifies contract repapering and back-office changes as major implementation burdens [DTCC et al., 2025]. Finally, Aronoff and Rajan [2026] use granular data on US Treasury repo trades to show that spreads on repo chains where intermediaries clear trades are elevated compared to other chains.

The design objective for NetWrap address the key shrotcomings of incumbent netting methods. A mechanism should capture funded multilateral netting without requiring exposure rewiring, CCP substitution, or all-or-nothing failure. That requires four properties: settlement only against verified resources; local resolution of funding shortfalls; bilateral reinstatement of unfunded portions between the original counterparties; and an auditable recovery path that identifies which portions settled, which portions were peeled out, and where residual exposure resides. Section 3 introduces NetWrap as a wrapper that implements these properties while leaving the choice of netting core modular.

3 Elements of NetWrap

The previous section distilled the design problem implied by incumbent netting cores: settlement must remain correct under funding shortfalls without reallocating counterparty exposure or relying on discretionary operator intervention. NetWrap addresses this by wrap-

ping a chosen netting core with a small set of protocol primitives that (i) require verifiable pre-funding of net outflows, (ii) recover from shortfalls by removing only deficiency-linked portions and reinstating them bilaterally, and (iii) record deterministic, auditable state transitions in a verifiable transcript. The remainder of this section introduces the mechanism operator and the wrapper primitives at an architectural level; Section 5 provides the formal step-by-step specification and algorithms.

3.1 The Mechanism Operator (wrapper controller)

NetWrap is coordinated by a *mechanism operator* (MO): a computing process that implements the wrapper control logic around a chosen netting core. The MO is not a trading counterparty and does not assume the credit exposure created by directed obligations among agents. Its role is to execute a deterministic workflow: ingest committed initial directed obligations, form netting groups and replace directed obligations with multiparty coordination records ("MCRs"), coordinate lock/verify commitments, apply deficiency-linked removal and re-netting when shortfalls occur, and trigger settlement only for fully funded netting groups.

Operationally, the MO is realized as one or more servers and/or smart-contract components that exchange authenticated messages with nodes and custody/ledger systems. The MO maintains the protocol state as an event-sourced state machine and emits an authenticated transcript of state transitions sufficient for replay and audit. The specific realization of confidentiality and trust (e.g., TEE execution, distributed operator nodes, or proof-based verification) is an optional implementation variant and does not change the formal protocol steps in Section 5 (see Section 6.3 and Appendix E).

Operationally, MO control messages are run-bound and version-aware. Commit instructions specify the object type, required quantity, commitment window, and run_id; each MCR generation or cancellation carries a version identifier; and completion-phase settlement instructions reference only the locked objects associated with the terminal MCR for that run. This prevents duplicate settlement of superseded assignments and allows replay logic to suppress stale or repeated messages.

MO responsibilities. At a minimum, a conforming MO must

(1) retain the protocol-relevant contract and assignment state,

(2) compute net directed obligations and issue/cancel replacement contracts per the stated rules,

(3) verify and record lock/commitment receipts,

(4) apply the rationing and deficiency-removal rules to produce residual netting groups and recovered bilateral directed obligations, and

(5) authorize or trigger settlement instructions once funding conditions are met.

3.2 NetWrap primitives

NetWrap is a protocol wrapper that can be layered over a chosen netting core. Rather than changing the arithmetic of netting, NetWrap adds three control primitives that govern when netted directed obligations may execute, how shortfalls are handled, and how execution is made verifiable. Section 5 provides the formal step-by-step specification; here we state the salient primitives at the architectural level.

Primitive 1: Lock–Verify pre-funding (commitment before settlement). Before any netted directed obligations are executed, each node with a net outflow must place the required quantity of the relevant object(s) into a verifiable locked state (escrow or smart-contract lock) and provide a receipt/proof bound to the protocol instance. This converts the chosen netting core from an all-or-nothing execution that is vulnerable to last-minute shortfalls into a pre-funded workflow in which settlement consumes only previously locked objects.

Primitive 2: Deficiency-linked removal with bilateral reinstatement (partial recovery). If a node cannot lock its full required net outflow, NetWrap does not abort the entire netting instance. Instead, it removes only deficiency-linked portions of that node's sending assignments, issues bilateral contracts for the removed portions between the original initial-contract counterparties, and re-nets the residual assignments. This preserves the exposure graph for unfunded portions (counterparty risk does not migrate to new counterparties) while allowing the remaining funded directed obligations to clear.

Primitive 3: Bounded iteration with a verifiable transcript (deterministic, auditable execution). The commit–verify and remove–re-net steps repeat deterministically until all remaining netting groups are fully funded, at which point settlement proceeds. Each state transition appends an authenticated record to a hash-chained transcript, enabling replay, integrity checks, and selective audit (with optional privacy mechanisms described in Section 6.3 and Appendix E). The result is a finite, protocol-level recovery path from shortfalls that is verifiable without trusting discretionary operator reports.

3.3 Implementation interfaces and required components

Section 5 specifies NetWrap as an abstract protocol over contracts, messages, traded objects, locking, and verifiable receipts. Implementations instantiate these abstractions via a small set of interfaces and supporting components; the protocol is agnostic to the underlying rail (institutional ledgers or smart contracts) provided the interfaces below are satisfied.

Core interfaces. A conforming implementation provides:

(i) a custody/locking interface Lock that places traded objects into a verifiable reserved state and returns a receipt/proof bound to the protocol instance;

(ii) a settlement interface Settle that executes the transfer instructions implied by a fully funded MCR and returns execution receipts; and

(iii) a logging interface that appends authenticated state-transition records to a tamper-evident transcript sufficient for replay and audit.

Supporting components Implementations typically include: authenticated message transport between nodes, the MO, and custody rails; a deterministic compute environment for Steps 3–8; and durable storage for protocol state and the transcript. Confidentiality and trust hardening (e.g., TEE execution, MPC/ZK proofs, or distributed operator nodes) are optional implementation variants that do not change Steps 1–9 (see Section 6.3 and Appendix E).

A conforming implementation also maintains explicit persistent state for at least: (i) an assignment table mapping directed obligation portions to coordination groups and retained scale factors, (ii) a commitment ledger of verified locks, (iii) a versioned MCR store, (iv) a deficiency graph or log of reinstated bilateral leftovers, and (v) a transcript ledger of authenticated event records. Commit and completion-phase messages may carry digital signatures, timestamps, and unique message identifiers so that replay after interruption is idempotent and duplicate processing is suppressed.

3.4 Summary of effects and roadmap

NetWrap enables multilateral netting while preserving the economic content of the initial contract set: funded portions settle via MCR net directed obligations, and any unfunded portions are removed and reinstated as bilateral directed obligations between the original counterparties before residual re-netting. As a result, settlement volume can be reduced without introducing new counterparty exposures or requiring a central guarantor to mutualize default risk. The protocol's formal properties—including exposure-graph preservation,

maximal settlement of committed traded objects, and bounded termination with a verifiable transcript—are stated in Section 5.1 and proven in Appendix D.

4 The Inner Loop: Deficiency, Removal, and Re-netting

This section presents a worked example of the inner loop: re-netting after the protocol removes the portions of assigned trades associated with a funding deficiency. For simplicity, we illustrate the process using a single coordination group. Table 1 lists the initial directed obligations assigned to the coordination group. These are the "directed obligation assignments." There are two traded objects, y and x, and six nodes $\{A, B, \ldots, F\}$. Each column is a bilateral assignment, shown by its two counterparties, with the traded-object terms suppressed in the header. In Tables 1, 3, 7, 8, and 9, each cell entry is expressed relative to the second named node in the column header $\{U, V\}$: "In N" means that V receives N units of the traded object from U, and "Out N" means that V sends N units of the traded object to U. For example, in the column headed by $\{A, B\}$, "In 2" in row x means that node B receives two units of x from node A, and "Out 9" in row y means that node B sends nine units of y to node A.

	{A,B}	{B,C}	{C,D}	{D,B}	{B,F}	{F,E}	{B,E}
x	In 2	In 2	In 2	Out 2	In 4	In 1	In 1
y	Out 9	Out 9	Out 4	In 7	Out 12	Out 2	Out 4

Table 1: Initial directed obligations assigned to a single coordination group

Table 2 shows the result of netting the traded-object flows in Table 1. In this table, "In N" indicates a net inflow of N units to the named node, and "Out N" indicates a net outflow of N units from the named node. For example, node F has a net outflow of 10 units of y before deficiency removal.

	A	B	C	D	E	F
x	Out 2	Out 3			In 2	In 3
y	In 9	In 9	Out 5	In 3	Out 6	Out 10

Table 2: Resulting netted directed obligations for each node and object type

Suppose node F commits only 4 units of y. Because Table 2 requires F to send 10 units of y, F has a deficiency of 6 units of y. The objective is to maximize netting among the other nodes while preserving bilateral exposure for the unfunded portion. This is achieved by removing from the coordination group the portion of the directed obligation assignment in which F is obligated to send y in an amount equal to the deficiency. In this example, the

removed portion is one-half of the $\{B, F\}$ assignment: F's 12-unit y obligation is reduced by 6, and the proportional 2-unit x leg is removed with it. The removed portion, $\{B, F; 2, 6\}$, is reinstated as a bilateral contract between B and F. The remaining trades form the residual coordination group shown in Table 3.

	{A,B}	{B,C}	{C,D}	{D,B}	{B,F}	{F,E}	{B,E}
x	In 2	In 2	In 2	Out 2	In 2	In 1	In 1
y	Out 9	Out 9	Out 4	In 7	Out 6	Out 2	Out 4

Table 3: Residual coordination group after partial removal

Table 4 shows the netted directed obligations for the residual directed obligations shown in Table 3. As discussed above, the deficient portion is removed from the MCR, which is canceled, and a new MCR is generated for the residual coordination group.

	A	B	C	D	E	F
x	Out 2	Out 1	In 0	In 0	In 2	In 1
y	In 9	In 3	Out 5	In 3	Out 6	Out 4

Table 4: Netted directed obligations for the residual coordination group

Comparing Table 4 with Table 2, only B and F change their directed obligations after the removal of one-half of the $\{B, F\}$ directed-obligation assignment. For all other nodes, the net amount of each traded object sent or received is unchanged by the re-netting. This reflects a general property that is stated in Proposition 1 in Section 5.1.

Appendix A extends the running example in two directions: (i) dividing initial directed obligations across multiple coordination groups and (ii) comparing the outcome to central clearing of the initial directed obligations.

5 The NetWrap Protocol

In this section we state NetWrap as a formal protocol: it defines what a correct execution must do, independent of any particular technology stack. The protocol is written in terms of abstract interfaces—contracts, messages, traded objects, escrow/locking, and verifiable receipts—that can be instantiated in either institution-based infrastructures (e.g., regulated accounts) or blockchain-based infrastructures (e.g., on-chain assets and smart contracts). The numbered steps below specify required inputs, state transitions, and outputs, while later sections and appendices discuss implementation choices and audit/privacy options.

Throughout, Section 5 is intended to be read as the canonical protocol description; the implementation material is explanatory.[4]

The numbered steps below specify a canonical NetWrap execution and the required interfaces (Lock, Ration, Peel, Settle, Audit) for a correct run. The protocol can also be executed with any other implementation that realizes the same lock–verify–settle outer loop and deficiency-removal inner loop using these interfaces, including equivalent combinations or decompositions of steps that preserve the protocol invariants.

The protocol executes as two nested loops: an **outer-loop** that processes a batch of trades from assignment through commitment and settlement, and an **inner-loop** that selectively reprocesses only those portions that fail commitment. In the outer loop, the mechanism operator (MO) takes the initial coordination group, partitions it into directed obligation assignments and coordination groups (Step 3), computes each node's net flows (Step 4), replaces the initial trade contracts with multiparty coordination records (MCRs) and gathers nodes' priority orderings (Step 5), and then verifies and commits each node's aggregate MCR net outflow (Step 6); once all groups are fully committed, the MO executes the Object Sending Algorithm to deliver the traded objects (Step 9). If any node cannot commit its full net outflow, the inner loop is triggered: the Deficiency Removal Algorithm removes from the deficient node's sending trades a portion that includes the amount of its deficiency, issuing bilateral contracts for the removed assignments and leaving a residual coordination group (Step 7); the MO then cancels the prior MCRs (retaining earlier commitments) and re-applies the assignment algorithm (Step 3) to each residual group, repeating netting and commitment until no deficiencies remain (Step 8).

Definitions and Terminology

The following terms are used consistently in the protocol description and proofs. They provide the operational meaning needed to state the algorithms and the properties proved below.

node A node that is a counterparty to one or more initial scheduled transfers and interacts with the Mechanism Operator (MO) during protocol execution.

object A quantified digital asset, reservable entitlement, or ledger entry verifiable on an independent system (e.g., bank ledger, smart contract, quota manager, or other custody

[4] A working Python implementation and simulation of Steps 1–9 are available at `https://github.com/DanielAronoff/netwrap`. The repository demonstrates execution traces for both institutional and blockchain custody rails and corresponds to the pseudocode herein. Appendix C describes the hardware, network, and cryptographic infrastructure that executes the state model.

rail). The term excludes informal promises or unrecorded off-ledger entitlements.

mechanism operator (MO) The computing process or node that executes Algorithms 1–9. The MO performs only computational and messaging functions and is never a contracting counterparty.

initial trade contract / initial trade A bilateral contract between two nodes specifying reciprocal transfers of one or more traded objects (direction and quantity). The set of all confirmed initial scheduled transfers submitted to the MO constitutes the *initial coordination group*.

directed obligation assignment An initial trade, or portion thereof, designated for inclusion in a specific coordination group according to the Directed Obligation Assignment Algorithm.

coordination group A finite, explicitly enumerated set of directed obligation assignments netted together in a single invocation of the Netting Algorithm. No directed obligation assignment belongs to more than one netting group within the same iteration.

multiparty coordination record (MCR) For each coordination group, the MO replaces associated initial scheduled transfers with an MCR that specifies each node's net directed obligations per traded object within that group.[5]

net outflow / net inflow For a node and traded object within an MCR, the net outflow is defined as [outflow − inflow]. A positive value is a net outflow; a negative value is a net inflow. A node's aggregate MCR net outflow for a traded object is the sum of its net outflows for that traded object across all MCRs.

escrow / lock / commitment The verifiable technical state in which a specified quantity of a traded object is reserved against reuse and evidenced by a receipt or on-chain proof. This condition must occur before settlement of the corresponding MCR.

protocol instance identifier (run_id) A unique identifier for a protocol execution that binds commitment receipts, MCR versions, deficiency events, and settlement receipts to a single run and is included in the transcript records.

message identifier / Idempotency Key A unique identifier attached to a commit instruction, completion-phase instruction, lock proof, or settlement receipt so that retries, late receipts, and replay events can be recognized as duplicates rather than applied twice.

[5] In any assignment protocol (including CCP-style assignment), bilateral contracts issued for peeled portions name the initial-contract counterparties, not any intermediate bookkeeping identifier.

commitment deadline / window A time bound during which a node must produce a verifiable commitment receipt covering its required aggregate MCR net outflow for a traded object; failure to do so results in a deficiency for purposes of Step 7.

fully funded (or fully committed) coordination group A coordination group for which all required outflows are covered by verified locked amounts after applying the rationing rule (equivalently, no deficiencies remain for any node/traded object in that group).

deficiency A shortfall between a node's committed amount and its required aggregate MCR net outflow for a given traded object:

$$\delta_{a,k} = \max\big(0,\ \mathrm{ReqAgg}[a,k] - \mathrm{Locked}[a,k]\big).$$

Rationing Rule If a node's committed quantity is insufficient to fund all required outflows across netting groups, this rule specifies the priority ordering or allocation of committed traded objects.

deficiency assignment rule Within a deficient coordination group, specifies how the remaining deficiency is allocated across the deficient node's sending directed obligation assignments (e.g., by scaling factor $\alpha \in [0, 1]$).

outer loop / inner loop The outer loop performs assignment, netting, contract replacement, and verification of commitments. The inner loop removes deficiency-linked portions of trades, reinstates those portions bilaterally, and re-nets the residual groups until all deficiencies are eliminated.

minimum transferable unit For a given object type, the smallest transferable quantity permitted by the relevant custody rail (e.g., integer token unit, lot size, minimum denomination, bandwidth slice, or reservation slot). Where exact deficiency matching is impossible under such discreteness constraints, NetWrap removes the minimum quantity not less than the deficiency under a deterministic tie-breaking rule and records any overshoot in the transcript.

fixed run / rolling system The bounded-termination and bounded-resolution results in this paper are evaluated for a fixed finite dataset or queue snapshot with deterministic tie-breaking. Rolling liquidity-saving systems with continual arrivals can invoke NetWrap on successive snapshots, but the guarantee applies to each invocation rather than to the market as an indefinitely evolving whole.

Computational workflow In the reference workflow, the MO:

(i) receives and validates digitally signed initial-contract records that specify counterparties, traded objects, directed quantities, and a protocol commitment for the run;

(ii) partitions the validated records into directed obligation assignments and coordination groups under a specified assignment rule, and computes per-node net outflows per traded object within each group;

(iii) generates, issues, and stores a multiparty coordination record (MCR) for each coordination group, wherein each MCR specifies each node's net directed obligations per traded object, and cancels or suspends the corresponding assigned portions of the initial-contract records;

(iv) obtains or derives a priority ordering over coordination groups for allocating a node's limited committed traded objects (a rationing rule);

(v) for each node and each traded object, verifies availability and places the required aggregate net outflow into a verifiable locked state via a custody/escrow interface, and records a commitment receipt bound to the run;

(vi) determines any deficiency as a shortfall between required outflows and locked amounts, allocates locked amounts across groups under the rationing rule, and, for any deficient group, removes deficiency-linked portions of the deficient node's sending assignments;

(vii) issues bilateral contract records for peeled portions between the original initial-contract counterparties and reprocesses the residual assignments by repeating the assignment and netting steps;

(viii) repeats the assignment, netting, and commitment steps until all remaining groups are fully funded; and

(ix) executes settlement by instructing a settlement interface to transfer only locked traded objects to satisfy MCR net inflows, while appending authenticated state-transition records and receipts to a tamper-evident transcript for replay and audit.

Protocol steps

Step 1. nodes enter into contracts to trade two traded objects, y and x (the "initial scheduled transfers").

Step 2. nodes send initial scheduled transfers to the MO. Each initial trade contract specifies:

(a) the counterparties (A, B);

(b) the set of traded objects exchanged (for exposition, y and x);

(c) the direction and quantity of each traded object flow (e.g., $\{A, B; M, T\}$, where $A \to B : q^M_{A\to B}$ units of y and $B \to A : q^T_{B\to A}$ units of x), including any applicable settlement identifiers; and

(d) a commitment that both counterparties are bound to participate in the NetWrap protocol for that contract.

The MO confirms that each submitted contract is well-formed and contains the required commitment. If all contracts meet these criteria, the protocol moves to Step 3. The set of confirmed initial scheduled transfers is the initial coordination group.

> *Example (Steps 1–2: contract submission)*
> Suppose A and B enter an initial trade contract in which A sends 5 units of y to B and B sends 5 units of x to A, $\{A, B; 5, 5\}$. The contract submitted to the MO includes counterparties (A, B), the traded objects (M, T), the directed quantities (e.g., $A \to B : 5M$ and $B \to A : 5T$), and a statement binding both counterparties to NetWrap for this contract.

Step 3. The MO partitions the initial trades into directed obligation assignments, each with an assigned value of traded objects ("directed obligation assignment value") and assigns them to groups ("coordination groups") in accordance with the Directed Obligation Assignment Algorithm.

> *Example (Step 3: coordination groups)*
> Given an initial coordination group $\{t_1, t_2, t_3, t_4\}$, where t_i represents some initial trade contract $\{A, B; M, T\}$, the Directed Obligation Assignment Algorithm may assign trades to coordination groups G_1 and G_2, possibly after splitting a trade proportionately between y and x (e.g., $t_4 = t_{4a} + t_{4b}$ where $t_{4a} = \{A, B; \alpha M, \alpha T\}$ and $t_{4b} = \{A, B; (1-\alpha)M, (1-\alpha)T\}$) so that $t_{4a} \in G_1$ and $t_{4b} \in G_2$ while preserving each bilateral contract's total traded object flows.

Step 4. For each coordination group, the MO computes the net flows of traded objects for each node.

Example (Step 4: net outflow computation)
In coordination group t_i, if node A has total outflow of 8 units of y and total inflow of 3 units of y across its assigned trades, its net outflow of y is $(8-3)=5$. If its outflow of x is 2 and its inflow of x is 7, then its net outflow of x is $(2-7)=-5$; i.e., a net inflow of 5 units of x.

Step 5. The MO notifies nodes that the assigned portions of the initial scheduled transfers are canceled or suspended for settlement and replaced by a new MCR for each coordination group. Each MCR carries a version identifier and specifies each node's net directed obligations per object. Each node messages the MO with a priority ordering (ranked list) of coordination groups to apply its committed traded objects. Marking assigned portions as replaced or suspended prevents double settlement while the run is in progress.

Example (informative; Step 5: priority ordering across coordination groups)
If a node participates in two coordination groups G_1 and G_2, it may message the MO with a priority ordering (ranked list) such as $G_1 \succ G_2$, indicating that any committed traded objects should be allocated to satisfy directed obligations in G_1 before G_2 under the Rationing Rule.

Step 6. MO applies the COMMIT–VERIFY ALGORITHM to (i) verify whether a node possesses its aggregate MCR net outflow and (ii) commit the verified traded object(s) to the protocol. Verification is performed by obtaining an authoritative receipt or on-chain proof from the relevant custody/ledger system that the specified quantity is reserved against reuse and bound to the run. The commit instruction includes at least the run_id, object identifiers, required quantity, and a commitment window; commitment receipts may further carry digital signatures, timestamps, and unique message identifiers to support idempotent replay and duplicate suppression.

Example (Step 6: locking receipt)
In an institutional implementation, a lock/commitment may be represented by a receipt (e.g., an attested message) stating "node A locked x units of y in escrow at time τ," along with identifiers binding the receipt to the relevant protocol instance and (optionally) to the set of coordination groups covered by that commitment.

Step 7. For each MCR for which one or more nodes do not commit their full required MCR net outflow in Step 6, the MO applies the DEFICIENCY REMOVAL ALGORITHM. This partitions the affected directed obligation assignments into two categories: (i) residual coordination groups (to be reprocessed) and (ii) bilateral trades (to be issued as bilateral contracts). MCRs (or coordination groups) for which all required commitments are satisfied in Step 6 advance directly to Step 9 for settlement; deficient MCRs do not advance to settlement until their deficiencies have been removed and the residual groups have been reprocessed under Step 8.

Step 8. MO cancels superseded MCR versions, while retaining the commitments in Step 6 to the extent they remain applicable, and

- issues contracts for each bilateral trade, and
- reprocesses each residual coordination group by treating its residual directed obligation assignments as the next-round trade set and repeating Steps 3–6 (assignment, netting/MCR generation, and lock–verify) until the residual coordination group becomes fully funded.

> *Example (Step 8: residual re-netting)*
> After Step 7 produces a residual coordination group G', the MO treats the remaining directed obligation assignments in G' as a coordination group for purposes of Step 4, recomputes net flows and replacement contracts, and repeats lock–verify until the residual coordination group becomes fully committed.

Note: Finite iterations. Each iteration of Steps 6–8 either (i) identifies a fully committed set of netting-group directed obligations that can advance to settlement, or (ii) removes a positive portion of deficient sending assignments (reinstating them bilaterally), thereby strictly shrinking the residual problem. In discrete-unit settings the shrinkage is measured in minimum transferable units, and any deterministic tie-breaking outcome is recorded in the transcript. The bounded-termination result therefore applies to a fixed finite run or snapshot.

Step 9. MO initiates the sending of traded objects in accordance with the OBJECT SENDING ALGORITHM. Completion-phase settlement instructions reference the terminal MCR version and the run_id and cause transfers only from previously locked objects. Settlement receipts are appended to the transcript and may be selectively disclosed, or proved in zero knowledge, to auditors.

Step 10. *Optional.* An authorized auditor views all or part of the transcript in accordance with the AUDITING ALGORITHM.

> *Example (Step 10: audit query)*
> An authorized auditor may query "Which nodes were deficient in traded object y during this protocol instance?" The MO can answer with the set of deficient nodes and (if required) provide a verifiable proof/attestation that the answer is consistent with the transcript, without disclosing unrelated position-level details.

Algorithms

DIRECTED OBLIGATION ASSIGNMENT ALGORITHM

All initial scheduled transfers must state rules for assigning trades to coordination groups and for rationing committed traded objects among those groups. The rule can include node input at Step 5. Any such rule must satisfy the invariant constraints below. Subject to those constraints, the protocol is agnostic about the selected assignment rule, provided that the same rule is stated in each initial trade contract.

I. *Assignment Constraints*: The MO rearranges the initial coordination group in accordance with a rule for assigning trades, which is stated in the initial scheduled transfers (the "assignment protocol"). It can be any algorithm that partitions the traded object flows between counterparties into one or more trades that are assigned ("directed obligation assignments") to groups of partitioned trades ("coordination groups") subject to the following constraints:

i each directed obligation assignment is located in one coordination group.

ii for each traded object in each initial trade contract, the sum of the traded object flow volume between counterparties that is assigned to coordination groups equals the traded object flow in the initial trade contract.

iii any volume of the traded object that is not assigned to a coordination group is returned (or released) to the node.

The rule for assigning trades is the Directed Obligation Assignment Rule.

When the assignment protocol is central clearing, the following central-clearing assignment algorithm is used.

CENTRAL CLEARING ALGORITHM

Each initial trade, for example between nodes A and B, is novated and replaced by two trades. One trade is between A and the central clearing counterparty ("CCP"), where the CCP assumes the position of B. The other trade is between the CCP and B, where the CCP assumes the position of A. The set of novated and replaced trades forms a single coordination group.

> *Example (Central clearing directed obligation assignments)*
> Suppose the initial trade involves A sending 5 units of y and B sending 5 units of x, $\{A, B; 5, 5\}$. The contract is novated and replaced by two contracts: $\{A, CCP; 5, 5\}$ and $\{CCP, B; 5, 5\}$.[a]
>
> [a] A more detailed example of central clearing is provided in Appendix A.1.

Clarification (CCP-style assignment under NetWrap). When Central Clearing is used as an assignment protocol within NetWrap, the CCP identifier is a bookkeeping hub for forming net directed obligations, not a guarantor that absorbs default exposure. Any MCR leg that is not fully funded in Step 6 is peeled in Step 7 and reinstated as a bilateral directed obligation between the original initial-contract counterparties for at least the deficiency amount. Accordingly, unfunded portions revert to the original bilateral exposure graph rather than migrating to the CCP.

COMMIT–VERIFY ALGORITHM

Two cases:

- *Nodes hold traded objects in accounts at a financial institution.* Each node authorizes the MO to instruct the financial institution to verify and commit its traded objects in accordance with the protocol.
- *Nodes initially hold traded objects in a blockchain wallet or smart-contract account.* The MO instructs nodes to send the required traded objects to escrow, where the MO verifies receipt and locks the traded objects.[6]

Optional regulator backstop. The MO may notify an authorized regulator or backstop facility of deficiencies. The regulator or facility may fund the deficiency amount into the relevant node or escrow account, after which the funded amounts are committed. Any liability between the deficient node and the regulator or facility is resolved outside the protocol.

DEFICIENCY REMOVAL ALGORITHM

[6] If the traded objects are held in a smart contract, e.g., if y is a stablecoin, then the arrangement may be that nodes authorize the MO to instruct the smart-contract operator.

When a node is deficient in a traded object (i.e. fails to commit the full amount of its aggregate MCR net outflow of that traded object):

First, the node's committed traded objects are allocated among MCRs in accordance with the Rationing Rule.

Second, for each MCR where the node remains deficient in the traded object, the amount of deficiency is allocated among the deficient node's directed obligation assignments to the affected MCR (made in Step 3 of the protocol), such that the deficient node's directed obligation assignments are scaled down by the amount necessary to cure its deficiency.[7]

The rule works as follows. Let t_i be a directed obligation assignment to MCR G_j involving node A, and suppose A has a deficiency in the traded object y associated with that assignment. Let $\alpha \in [0, 1]$ denote the retained scale factor. The retained portion $\alpha(t_i)$ remains in coordination group G_j, while the removed portion $(1 - \alpha)t_i$ is peeled from the group to cure the deficiency. A new bilateral contract is issued for each removed directed obligation assignment. The residual coordination group G'_j is the set of directed obligation assignments that remains after the removed portions have been peeled away.

In discrete-unit implementations, each retained portion $\alpha(t_i)$ and each removed portion $(1 - \alpha)t_i$ must be an integer multiple of the relevant minimum transferable unit. If exact equality between the peeled quantity and the deficiency is infeasible, the coordinator chooses the smallest peeled quantity not less than the deficiency under a deterministic tie-breaking rule, and records both the overshoot and the tie-breaking outcome in the transcript. When an assignment references multiple object types, peeling is applied object by object, so a deficiency in one object need not force removal of a non-deficient object.

> *Example (partial fulfillment of deficiency)*
> A has a deficiency of 2 units of y in coordination group G_j. A's directed obligation assignments to the coordination group are: $t_1 : \{A, B; 2, 2\}$, $t_2 : \{A, C; 3, 3\}$, $t_3 : \{A, D; 1, 1\}$ and $t_4 : \{E, A; 3, 6\}$. A is obligated to commit 3 units of y after netting. The requirement is to remove from the coordination group portions of A's directed obligation assignments that involve 2 units of y. One deterministic way is to remove t_3 and half of t_1 from the directed obligation assignments. The resulting directed obligation assignments are $t'_1 : \{A, B; 1, 1\}$, $t_2 : \{A, C; 3, 3\}$, and $t_4 : \{E, A; 3, 6\}$. The recovered bilateral trades are $t''_1 : \{A, B; 1, 1\}$ and $t_3 : \{A, D; 1, 1\}$.

The peel rule also operates object by object unless the chosen assignment protocol treats a

[7] An alternative is to remove all of the deficient node's directed obligation assignments to the affected MCR.

multi-object instruction atomically. Accordingly, a deficiency in one object need not force removal of non-deficient objects, and different deterministic peel rules can be substituted without altering the wrapper's lock–verify–re-net structure.

RATIONING RULE

When a node is deficient in a traded object, its committed traded objects are assigned to coordination groups subject to the following constraints:

i The traded object is assigned to coordination groups in an order of priority provided by the node (i.e., first to coordination group G_j, then to coordination group G_k, etc.).

ii The assignment can either be made by the node when the deficiency occurs (in Step 7) or coded into the protocol. In the latter case priorities are based on criteria that can be applied to all possible configurations of coordination groups (e.g., by ascending or descending order of the volume of the traded object the node must send, or by the number of nodes in the coordination group, etc.).

OBJECT SENDING ALGORITHM

The MO routes traded objects from escrow to nodes to fulfill their net inflows. There are two cases to consider.

Case 1: traded objects held by institutions.[8] The MO sends an instruction to each institution to send traded objects to node accounts at the home institution and to other institutions where nodes hold accounts. The quantity of each traded object in escrow(s) matches the net inflows to nodes by definition. Any protocol is compatible with any algorithm that accomplishes this.

Example (routing of traded objects to nodes with institutions)
The MO instructs each institution to send traded objects in sequential order:

i Transfer traded objects to nodes with accounts at the institution up to each node's MCR net inflow. The remaining un-transferred traded objects are the institution's excess. The remaining aggregate MCR net inflows are the institution's deficit.

ii Generate a map of traded object flows from excess institutions to deficit institutions such that (i) all excesses are sent and (ii) traded objects received equal deficits.

iii Each institution transfers traded objects to nodes with remaining MCR net inflows.

[8] See worked example in Section 6.1.

Case 2: traded objects held in a smart-contract escrow.[9] The MO sends traded objects from escrow to nodes in the amount of their net inflows.

AUDITING ALGORITHM

First, an authorized auditor sends a message to the mechanism operator requesting information on the transaction transcript. The message contains verifiable credentials.

Second, the mechanism operator responds to the request.

(i) If the request is for verification that some element of the transcript satisfies certain properties, the MO initiates a zero-knowledge proof, either interactive or non-interactive, to prove whether the properties are satisfied.

(ii) If the request is to view the whole, or part, of a transcript, the mechanism operator sends an encrypted ciphertext with the requested transcript to the auditor.

An "authorized auditor" is either a legally empowered entity, such as a regulator or court of law, or a node authorized by a node to view its portion of the transaction transcript. The authorized auditor is sent a public/secret key pair to enable it to decrypt the transcript ciphertext it is sent.

Note (minimal transcript). A sufficient transcript for auditing can include: (i) identifiers/hashes of initial scheduled transfers; (ii) the trade-assignment mapping and netting-group membership; (iii) MCR versions and cancellation/replacement events; (iv) lock/commitment receipts; (v) deficiency notices and scaling/remove records; and (vi) settlement instructions/execution receipts.

[9] See worked example in Section 6.2.

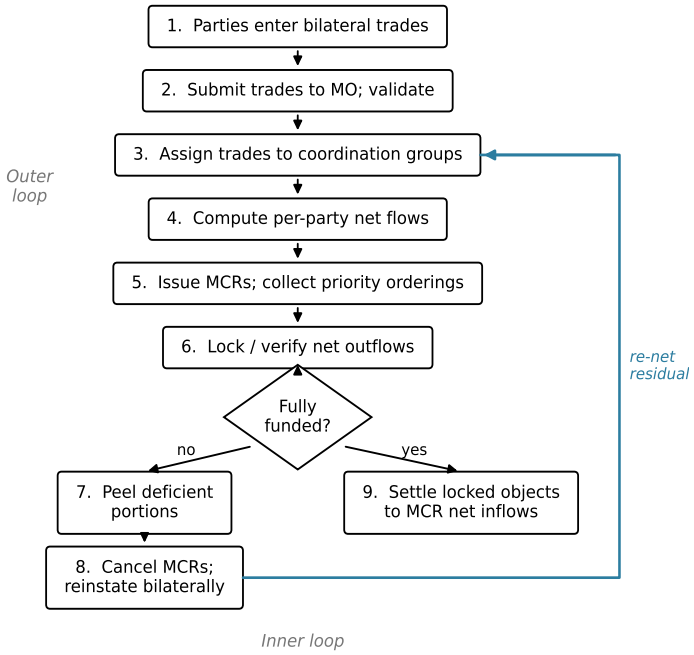

Figure 2: NetWrap Protocol

5.1 NetWrap Properties

Here we state propositions for key properties of the NetWrap protocol; maximal multilateral netting, profit invariance and counterparty risk preservation, and preservation of initial counterparty risk. The proofs are in Appendix D.

Proposition 1. (Netting of unrelated directed-obligation assignments unaffected by deficiency) *When there is a re-netting after removing directed obligation assignments from a coordination group, the only nodes that can experience a change in traded objects sent or received are counterparties in the removed directed obligation assignments.*

Corollary (Persistence of counterparty risk)**.** *When there is a re-netting after removing directed obligation assignments from a coordination group, the deficient node's original counterparty remains the counterparty for at least the amount of the deficiency.*

Proposition 2. (Maximal multilateral netting of committed traded objects) *The NetWrap protocol achieves maximal multilateral netting of committed traded objects.*

Proposition 3. (Profit invariance and counterparty-risk preservation) *Under the NetWrap protocol:*

(1) A node's contractual profit (i.e., profit under full performance of contractual directed obligations) is the same as under its initial scheduled transfers.

(2) A node's counterparty risk remains with its initial trade contract counterparties (i.e., the node is not exposed to default in a contract it did not initially enter).

Proposition 3A. (Adjacency invariance) *Let $G_0 = (V, E_0)$ denote the exposure graph formed by the set of initial contracts, and let E_t be the edge set after any iteration t. Then $E_t = E_0$ for all t.*

Proposition 4. (Deterministic finite termination and computational bound) *For any execution of the NetWrap protocol (Algorithms 1–9) on a digital computing system with a finite number of nodes N and indivisible traded object units, the protocol terminates after at most V inner-loop iterations and within a bounded wall-clock time*

$$T = \mathcal{O}(V\,\tau \log N),$$

where τ is the mean network round-trip latency. Each iteration appends a verifiable, cryptographically signed transition record to the system transcript $\mathcal{L}$.

6 NetWrap Implementation Maps

Two implementation maps illustrate deployment on:

(i) an institutional rail interfacing with bank APIs, and

(ii) a blockchain rail executing through smart contracts for ERC-20 or other smart-contract tokens.

In both cases, the state transitions in Steps 1–9 can be implemented with existing software stacks. Additional optional implementation variants instantiate the same interfaces under alternative privacy, governance, and performance assumptions without altering Steps 1–9 (Appendix E).

6.1 Traded objects are held at financial institutions

Figure 3 displays a map of NetWrap when parties hold y and x in accounts at financial institutions and those institutions hold the traded object escrows. The arrows represent messaging links. Each number corresponds to a protocol step and the location where it is executed (including the algorithms called by the step). The medium for sending messages can be pluralistic and include internet, private secure electronic channels and blockchains. We briefly comment on selected elements. The messaging between the regulator and other nodes, labeled as Step 6, reflects the optional element in the Verification and Commitment Algorithm. The sequence is as follows. First, the MO notifies the regulator of a deficiency.[10] Second, the MO may notify the institution with an instruction to cure the deficiency by moving traded objects from its account into escrow.[11] Third, the regulator messages the MO and the deficient parties that it has cured the deficiency. To implement this part of the protocol requires contractual or statutory rules that specify the authority of the regulator to cure the deficiency and the legal directed obligations of the parties. For example, the regulator may become the recipient of the traded objects sent to the deficient node. The messaging between the MO and the financial institutions, labeled as Steps 6 and 9, reflect the verification of locked traded objects from the institution to the MO and the instruction for release of traded objects from escrow to other financial institutions and parties from the MO to the institution. The semi-circular messaging between institutions, labeled as Step 9, reflects the movement of traded objects as depicted in Figure 5. Finally, the messaging between the MO and auditor, labeled as Step 10, reflects the query and response to auditor questions.

[10] the MO may also notify the regulator with other information, such as concentration of trading positions among parties.

[11] For example, the regulator may transfer central bank reserves to cover a deficiency in y and central bank holdings of Treasuries to cover a deficiency in$x$.

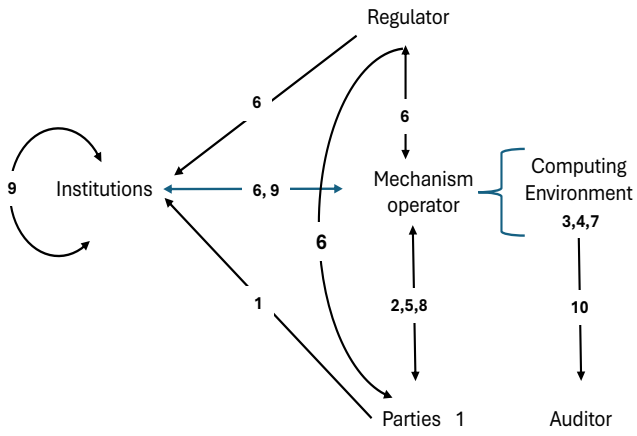

Figure 3: Protocol map: traded objects held at financial institutions

Figure 4 maps an MO server implemented as a TEE. Encrypted messages are received, decrypted, processed, re-encrypted, and sent onward. An encrypted transcript of the computation is generated and retained in the enclave.

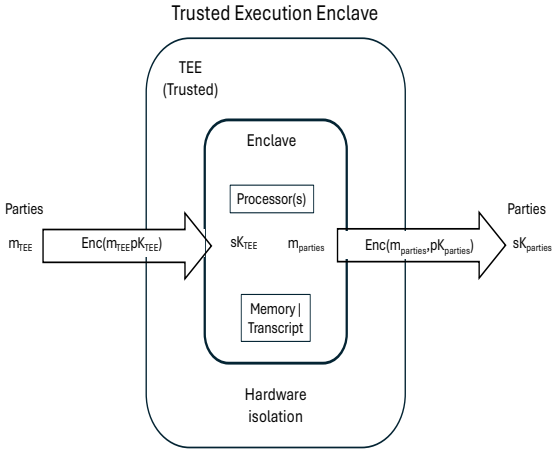

Figure 4: Trusted Execution Enclave

Figure 5 shows the movement of traded objects in the residual netted directed obligations in Table 2 when the traded objects are held in accounts at two financial institutions. Within an institution, traded objects move between accounts owned by parties and an escrow account owned by the institution and controlled by the MO. Across institutions, traded objects move between institutional escrow accounts. The map shows movement of traded objects y and x inside and across institutions. Parties A, B, and C hold traded object accounts with the first

institution and parties D, E, and F hold traded object accounts at the second institution.

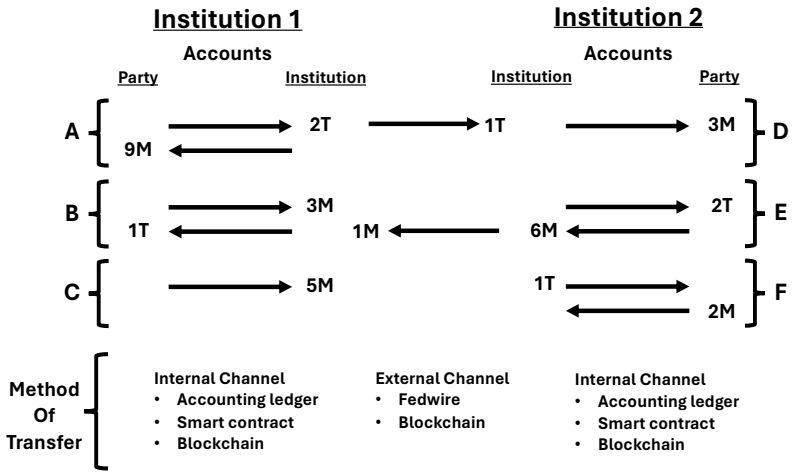

Figure 5: Residual netted directed obligations: movement of traded objects between institutional accounts

6.2 Traded objects are on blockchains

When traded objects are represented on a blockchain, the architecture of NetWrap is not fundamentally altered. The main technological difference is that smart contracts appended to blockchains replace financial institutions as the escrow and settlement rail. Here we describe NetWrap using the architecture of the ERC-20 standard for fungible tokens on the Ethereum Virtual Machine (see `https://ethereum.org/developers/docs/evm/`). NetWrap can be adapted to any Turing-complete blockchain that supports smart contracts. Figure 6 displays the map of NetWrap when y and x are account balances in a smart contract. Each traded object entry is identified by a public key, or by a hash of a public key. A node holds the associated private key in an off-chain wallet. The private key enables the node to initiate transfers of traded objects. As in the institutional implementation, a node can authorize the MO to transfer the traded object to the smart contract escrow. All other elements of the NetWrap blockchain protocol are the same as in Figure 3.

Inside the smart contract, balances move between node-owned accounts (on the left) and an MO-controlled escrow account (on the right). The MO instructs the smart contract to move balances and receives verification of transfers into the MO escrow. The state transitions of the smart contract, which register movement between accounts, are initiated by—and recorded on—blocks appended to the blockchain. Figure 6 consolidates y and x inside the same smart contract. Cross-rail and cross-chain instantiations are discussed in Appendix E.

Control of the smart contract. The smart contract does not need to be controlled by the

MO. For example, y can be a stablecoin represented as an account inside a smart contract controlled by the stablecoin issuer. The requirement is that the MO be given protocol-limited control over movement of the traded object among node accounts for the relevant run. In this case the MO depicted in Figure 6 is not the operator of the smart contract. A similar extension applies where a financial institution records traded objects, or account balances, on an internal or public blockchain.

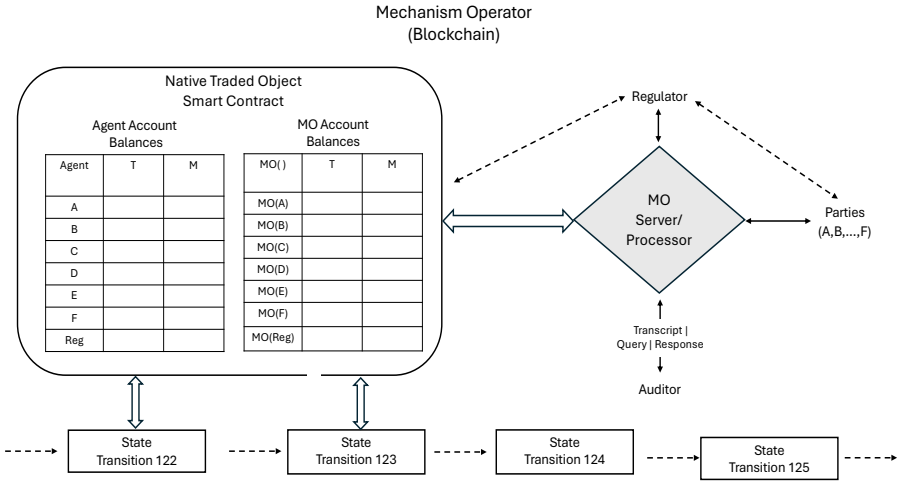

Figure 6: Protocol map: traded objects represented on blockchains

Figure 7 depicts the movement of traded objects in Table 2, the residual netted directed obligations, as transitions between node accounts and MO escrow accounts in a smart contract. State transition 1 shows the subtraction of traded object amounts from node accounts and their transfer to the MO escrow accounts. State 2 depicts the transfer of the traded objects from the escrow accounts—which reset to zero— to the node accounts.

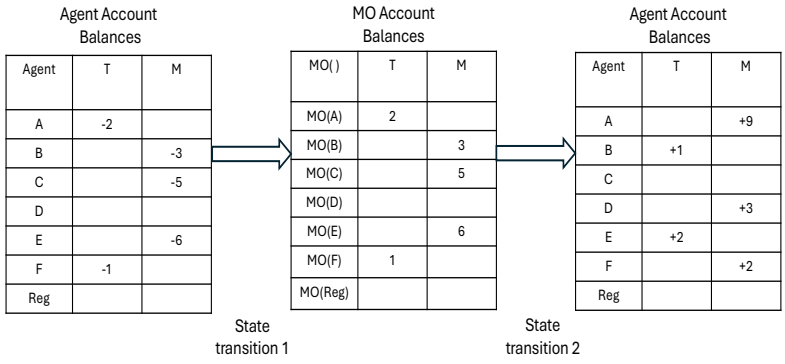

Figure 7: Residual netted directed obligations: movement of smart-contract balances

6.3 Implementation Variants: privacy, distributed control, and optimized scheduling

Section 5 is the *formal protocol description* of NetWrap; this subsection points to Appendix E, which records implementation variants that instantiate the same interfaces (Lock, Ration, Peel, Settle, Audit) without altering Steps 1–9.

- Privacy rails (TEE/MPC/ZK) (Appendix E.1).
- Distributed mechanism operator—threshold/BFT authorization (Appendix E.2).
- Optimized scheduling/parallel execution (Appendix E.3).

7 Conclusion

NetWrap is a fully specified and practically executable protocol wrapper that has been implemented in working code and verified through numerical simulation. It provides an implementable mechanism for reducing settlement risk while preserving counterparty exposures. The protocol requires participants with net outflows to lock the required traded objects in escrow before settlement. If any party cannot fund in full, NetWrap removes only the deficiency-linked portions of that party's assigned trades, reinstates those portions as bilateral directed obligations between the original counterparties, and re-nets the residual set. Iterating until no deficiencies remain ensures that the multilateral settlement that ultimately executes is fully funded, while leaving residual default exposure where it began. Equally importantly, the protocol is restart-safe: run-bound instructions, versioned replacement contracts, and a hash-linked transcript make the recovery path independently auditable and deterministic under replay.

By combining multilateral netting with counterparty-risk preservation and a deterministic, verifiable control structure, NetWrap occupies a design space that incumbent methods leave open. Trade compression and central clearing achieve netting but alter counterparty exposures or mutualize risk; cycle-based atomic-swap systems preserve exposures but terminate on the first failure. NetWrap's nested lock–verify–settle architecture provides the first synchronous, bounded-iteration workflow that preserves initial exposures, achieves maximal netting of funded directed obligations, and produces a cryptographically auditable execution trace. These properties constitute the paper's technical contribution to distributed settlement design.

Because the wrapper is defined in terms of verifiable locking, deficiency peeling, replacement-record versioning, and replayable transcript updates, the same architecture can be ported

across institutional rails, smart-contract rails, and non-financial reservation systems without changing the core economic logic.

For market participants, this design can make multilateral netting compatible with counterparty limits and reduce the need for additional hedging or margin that arises when exposure is reassigned. For regulators, NetWrap separates the informational benefits often associated with CCPs (timely knowledge of positions, funding, and deficiencies) from the mutualization of credit risk: a mechanism operator can report funding shortfalls and protocol outcomes without becoming the counterparty to every trade.

We demonstrated that NetWrap satisfies three key design attributes that make it feasible to implement in financial markets:

1. maximal multilateral netting;
2. profit invariance and counterparty risk preservation, and
3. low computational complexity.

Finally, NetWrap is compatible with multiple deterministic choices of (i) trade-assignment rules, (ii) rationing/priority rules across netting groups, and (iii) deficiency-allocation (peel) rules within a deficient group, provided the Assignment Constraints and the lock–verify–commit semantics in Section 5 are satisfied. These configuration choices affect throughput, privacy, and governance, but do not change the core protocol behavior: settlement executes only against verifiably locked outflows; any unfunded portions are peeled and reinstated between the original initial-contract counterparties; and the run is recorded as a replayable, authenticated transcript.

References

Daniel Aronoff, Robert M. Townsend, and Madars Virza. Repomech: A method to reduce the balance-sheet impact of repo intermediation, 2025. URL `https://arxiv.org/abs/2512.23842`.

Daniel Aronoff and Sriram Rajan. [TBD].

CLS Group. New fx compression service. Online news article, Jan 2015. URL `https://www.cls-group.com/news/new-fx-compression-service/`. CLS collaborates with TriOptima to deliver an FX forward compression service.

MarketsWiki. Trioptima. `https://www.marketswiki.com/wiki/TriOptima`, 2025. URL `https://www.marketswiki.com/wiki/TriOptima`. Accessed: 2025-12-31; information on

TriOptima, a provider of post-trade infrastructure and risk management services for OTC derivatives markets.

Monetary Authority of Singapore and Association of Banks in Singapore. Project ubin phase 2: Re-imagining interbank real-time gross settlement (rtgs) using distributed ledger technologies. Technical report, Monetary Authority of Singapore (MAS) and The Association of Banks in Singapore (ABS), Singapore, November 2017. URL https://www.mas.gov.sg/-/media/MAS/ProjectUbin/Project-Ubin-Phase-2-Reimagining-RTGS.pdf. Accessed 14 Nov 2025.

D'Errico, Marco, and Tarik Roukny. 2021. "Compressing Over-the-Counter Markets." *Operations Research* 69(6): 1660–1679. https://doi.org/10.1287/opre.2021.2107.

Duffie, Darrell. 2017. *Post-Crisis Bank Regulations and Financial Market Liquidity*. Thirteenth Paolo Baffi Lecture on Money and Finance, Banca d'Italia. https://www.bancaditalia.it/pubblicazioni/lezioni-baffi/pblecture-13/index.html.

Cochran, Paul, Sebastian Infante, Lubomir Petrasek, Zack Saravay, and Mary Tian. 2023. "Dealers' Treasury Market Intermediation and the Supplementary Leverage Ratio." *FEDS Notes*. Board of Governors of the Federal Reserve System, August 3. https://www.federalreserve.gov/econres/notes/feds-notes/dealers-treasury-market-intermediation-and-the-supplementary-leverage-ratio-20230803.html.

Bowman, David, Yesol Huh, and Sebastian Infante. 2024. "Balance-Sheet Netting in U.S. Treasury Markets and Central Clearing." Finance and Economics Discussion Series 2024-057. Board of Governors of the Federal Reserve System. https://doi.org/10.17016/FEDS.2024.057.

Copeland, Adam, and R. Jay Kahn. 2024. "Repo Intermediation and Central Clearing: An Analysis of Sponsored Repo." Staff Report No. 1140. Federal Reserve Bank of New York. https://doi.org/10.59576/sr.1140.

Hempel, Samuel J., R. Jay Kahn, Robert Mann, and Mark E. Paddrik. 2023. "Why Is So Much Repo Not Centrally Cleared?: Lessons from a Pilot Survey of Non-Centrally Cleared Repo Data." OFR Brief 23-01. Office of Financial Research, U.S. Department of the Treasury. https://www.financialresearch.gov/briefs/files/OFRBrief_23-01_Why-Is-So-Much-Repo-Not-Centrally-Cleared.pdf.

The Depository Trust & Clearing Corporation. 2023. "Looking to the Horizon: Assessing a Potential Expansion of U.S. Treasury Central Clearing." White

Paper, September. https://www.dtcc.com/-/media/Files/Downloads/WhitePapers/Accessing-Potential-Expansion-US-Treasury-Clearing-White-Paper.pdf.

Wuerffel, Nate. 2025. "Reassembly Required: Central Clearing Could Reshape the U.S. Treasury Market." BNY. Originally published November 8, 2023; updated February 26, 2025. https://www.bny.com/assets/corporate/documents/pdf/insights/central-clearing-us-treasury-market-reassembly-required.pdf.

The Depository Trust & Clearing Corporation, Securities Industry and Financial Markets Association, BNY, Broadridge, and The ValueExchange. 2025. "U.S. Treasury Central Clearing Pulse Survey: Key Findings." October. https://www.dtcc.com/-/media/Files/Downloads/Press-Room/VX-2025-10-US-Treasury-Central-Clearing-Key-Findings.pdf.

A Further Examples of Re-Netting

In this appendix we continue the analysis of Section 4 with inner-loop re-netting for two additional one-shot netting methods, central clearing and multiple coordination groups. Both examples start with the initial obligations in Table 1.

	{A,B}	{B,C}	{C,D}	{D,B}	{B,F}	{F,E}	{B,E}
x	In 2	In 2	In 2	Out 2	In 4	In 1	In 1
y	Out 9	Out 9	Out 4	In 7	Out 12	Out 2	Out 4

Initial directed obligations

A.1 Central clearing

Under central clearing initial contracts are novated and replaced by contracts with a central clearing counterparty ("CCP"). The CCP becomes the counterparty to each node in each directed obligation, as shown in Table 5.

	{A,CCP}	{B,CCP}	{C,CCP}	{D,CCP}	{B,CCP}	{F,CCP}	{B,CCP}
x	In 2	In 2	In 2	Out 2	In 4	In 1	In 1
y	Out 9	Out 9	Out 4	In 7	Out 12	Out 2	Out 4
	{CCP,B}	{CCP,C}	{CCP,D}	{CCP,B}	{CCP,F}	{CCP,E}	{CCP,E}
x	In 2	In 2	In 2	Out 2	In 4	In 1	In 1
y	Out 9	Out 9	Out 4	In 7	Out 12	Out 2	Out 4

Table 5: Gross directed obligations between each node and the CCP for each object type

Table 6 displays the netted directed obligations of each node. The CCP has no net directed obligations. This is a result from the nature of its intervention, where, for a given initial directed obligation, it sends and receives the same quantity of each object. The netted trades of each node are identical to the single coordination group case in Table 2. The only substantive effect of using Central Clearing as the netting core (as opposed to a single bilateral coordination group) is the bookkeeping representation of net flows through a CCP identifier; under NetWrap, unfunded portions are peeled and reinstated bilaterally, so residual default exposure remains with original initial-contract counterparties.

	A	B	C	D	E	F	CCP
x	Out 2				In 2	In 3	
y	In 9	In 2	Out 5	In 3	Out 6	Out 10	

Table 6: Netted directed obligations under central clearing

When node F commits only 4 units of y, the residual netting is the same as shown in Table 4. The difference is that the bookkeeping representation of the netting core routes net flows through a CCP identifier; however, under NetWrap any unfunded portion is peeled and reinstated as a bilateral directed obligation between the original initial-contract counterparties corresponding to the removed portion (e.g., restoring the appropriate portion of the original (B, F) relationship for at least the deficiency amount), while the residual set is re-netted and settled against locked assets.

A.2 Multiple coordination groups

This case assigns portions of the initial directed obligations to three separate coordination groups, as shown in Tables 7, 8 and 9.

	{A,B}	{B,E}
x	In 1	In 1
y	Out 4.5	Out 4

Table 7: Coordination group 1

	{A,B}	{B,F}	{F,E}
x	In 1	In 4	In 1
y	Out 4.5	Out 12	Out 2

Table 8: Coordination group 2

	{B,C}	{C,D}	{D,B}
x	In 2	In 2	Out 2
y	Out 9	Out 4	In 7

Table 9: Coordination group 3

Table 10 shows that the effect of creating separate coordination groups does not affect overall netting. The net directed obligations of each node are the same as in the netted single coordination group Table 2.

	A	B	C	D	E	F
Coordination group 1						
x	Out 1				In 1	
y	In 4.5	Out 0.5	In 0	In 0	Out 4	In 0
Coordination group 2						
x	Out 1	Out 3			In 1	In 3
y	In 4.5	In 7.5			Out 2	Out 10
Coordination group 3						
x						
y		In 2	Out 5	In 3		
Total						
x	Out 2	Out 3			In 2	In 3
y	In 9	In 9	Out 5	In 3	Out 6	Out 10

Table 10: Per-group netted directed obligations and totals across groups

Suppose that A locks 1 unit of x. By the Deficiency Removal Algorithm, 1 unit of A's directed

obligation to send x is removed from the netting and $1/2$ of the initial directed obligations $\{A, B; 2, 9\}$ is reinstated. The question is where to assign the 1 unit deficiency between coordination groups 1 and 2. This is determined by the Rationing Rule. For example, an equal allocation would change A's directed obligation assignment to (+0.5) units of x and (-2.25) units of y for each group. This, in turn, would alter the net directed obligations of A and B in coordination groups 1 and 2, while leaving the net directed obligations of other agents unchanged.

B Persistent State Model and Transcript Structure

This appendix defines the durable data model and verifiable transcript used by the NetWrap protocol. It specifies the logical state of the system—the data that is created, consumed, or updated by Protocol Steps 1–9—without prescribing the hardware or cryptographic primitives that enforce it (those are in Appendix C). The goal is to render NetWrap's workflow fully machine-interpretable and verifiable in polynomial time.

The following data structures constitute a complete specification of the information a computing system must maintain to execute the protocol. They can be implemented in any conventional database or distributed ledger and contain sufficient detail for a practitioner to reconstruct all protocol behaviors described in Section 5.

State-machine semantics

NetWrap operates as a deterministic, event-sourced state machine. The protocol state at iteration t is

$$S_t = \big(\mathcal{C}_t, \mathcal{A}_t, \mathcal{G}_t, \mathcal{M}_t, \mathcal{L}_t\big),$$

where $\mathcal{C}_t$ is the set of active contracts, $\mathcal{A}_t$ the party registry, $\mathcal{G}_t$ the set of netting groups, $\mathcal{M}_t$ the multiparty coordination records (MCRs), and $\mathcal{L}_t$ the current event transcript. Each algorithmic step transforms $S_t \to S_{t+1}$ by applying a pure function f_i whose inputs are explicit in the pseudocode.

Core data structures

Assignment table. A relational table mapping each initial trade contract to its netting-group assignment and scaling factor:

$$T_{\text{assign}} = \{(\texttt{contract\_id}, \texttt{group\_id}, \alpha, \texttt{version})\}.$$

Commitment ledger. Records all escrow instructions and receipts as immutable tuples

$$T_{\text{commit}} = \{(\texttt{party}, \texttt{object}, q, \texttt{timestamp}, h_{\text{commit}})\},$$

where h_{commit} is a cryptographic hash linking to the escrow proof.

Deficiency graph. A directed acyclic graph $D = (V, E)$ with V the parties and an edge (A, B, q_k) representing reinstatement of a bilateral obligation of quantity q_k from party A to party B after deficiency removal. The graph's adjacency set encodes counterparty-risk preservation and is hash-committed at each iteration.

Transcript ledger. Every state transition produces an event record

$$E_t = \langle \texttt{run\_id}, s_t, \text{op}_t, s_{t+1}, \tau_t, \sigma_{\text{MO}} \rangle,$$

whose hash $h_t = H(E_t \| h_{t-1})$ forms a Merkle chain. The transcript $\mathcal{L} = \{E_t\}$ is the single source of truth for audit and replay.

Persistence and replay

All tables and graphs are appended to tamper-evident storage. Checkpoint S_t is serialized as $\langle \texttt{run\_id}, h_t, R_t \rangle$, where R_t is the Merkle root of $\mathcal{L}$. To restore state after interruption, the mechanism operator replays $\mathcal{L}$ to R_t; this process is idempotent and does not duplicate settlements. In practice, each authenticated event record can include an event-type identifier, affected obligation or portion identifiers, the relevant MCR version, prior-hash linkage, timestamps, and digital signatures. Partitioning records also include retained scale factors and any discrete-unit tie-breaking outcomes, making the replay log sufficient to reconstruct the assignment table, commitment ledger, deficiency graph, and versioned MCR store.

Technical contribution

This persistent-state formalization enables:

(a) constant-time integrity checks and logarithmic-time proof verification;

(b) cryptographically provable invariants (object conservation, exposure-graph preservation, finite termination);

(c) deterministic, replayable recovery independent of market logic.

These properties make the protocol state explicit, replayable, and independently auditable for multilateral settlement.

Terminology consistency. The data-structure names defined in this appendix (Assignment Table, Commitment Ledger, Deficiency Graph, Transcript Ledger) correspond one-to-one with the algorithmic variables used in Section 5 and in the pseudocode. No other hidden state or implicit variable is assumed.

C Execution Architecture and Security Controls

This appendix describes the hardware, network, and cryptographic infrastructure that executes the state model defined in Appendix B. It specifies how the protocol steps can be executed securely and efficiently in a distributed implementation.

System composition

The NetWrap computing environment consists of:

- one or more **Mechanism Operator nodes** (MOs), each a server or virtual machine executing verified program code within a **Trusted Execution Environment** (TEE) or Hardware Security Module;
- authenticated network links between MOs and nodes using TLS 1.3 or QUIC with mutual attestation;
- optional regulator and auditor nodes with read-only access to the event transcript $\mathcal{L}$.

Execution flow

The MOs instantiate two nested state machines: an outer loop for contract ingestion, assignment, and settlement, and an inner loop for deficiency detection and re-netting. Each transition E_t is signed, appended to $\mathcal{L}$, and verified by peers before the next operation is scheduled. Concurrency across independent coordination groups $\mathcal{G}$ is achieved through **conflict-free replicated data types (CRDTs)** that guarantee deterministic merging without rollbacks.

Cryptographic and performance features

(a) **Signature and hash chain.** Every operation is digitally signed and linked by a hash chain, ensuring authenticity and immutability of $\mathcal{L}$.

(b) **Message authentication and duplicate suppression.** Commit and completion-phase instructions, and the corresponding lock proofs and settlement receipts, may carry digital signatures, timestamps, and unique message identifiers so that late or repeated messages are recognized as duplicates during replay or restart.

(c) **Bounded latency.** Each node operates under maximum round-trip τ; overall communication complexity is $\mathcal{O}(N \log N)$, improving on legacy $\mathcal{O}(N^2)$ message broadcasts.

(d) **Parallel execution.** Independent groups execute concurrently under deterministic scheduling, yielding measurable throughput gains in wall-clock performance.

(e) **Audit verification.** Validators can confirm correctness by verifying signatures and hash continuity in polynomial time, without re-running settlement logic.

Security and recovery

All cryptographic material (keys, nonces, attestations) is sealed within the TEE. State checkpoints from Appendix B are stored redundantly. If a node fails, replay of the verified transcript restores the exact pre-failure state without divergence.

Implementation properties

Executing NetWrap within this architecture provides the following implementation properties:

- reduced message and computation complexity;
- hardware-enforced atomic escrow locking;
- deterministic, verifiable termination within $\mathcal{O}(V\tau \log N)$ wall-clock time;
- cryptographic auditability of every operation.

These features specify the computational, messaging, and audit requirements of one distributed implementation architecture.

Implementation feasibility. The architecture uses standard servers, authenticated message channels, durable storage, and cryptographic controls. These components instantiate the protocol interfaces defined above; alternative secure-computing or custody/ledger implementations can satisfy the same interface requirements.

D Proofs of NetWrap Properties

In this appendix we prove the propositions stated in Section 5.1.

Proposition 1. (Netting of unrelated directed-obligation assignments unaffected by deficiency) *When there is a re-netting after removing directed obligation assignments*

from a coordination group, the only nodes that can experience a change in traded objects sent or received are counterparties in the removed directed obligation assignments.

Proof. Fix a coordination group G produced in Step 3 and its associated MCR produced in Step 5. For each traded object O_k (in the exposition, $O_k \in \{M, T\}$) and each node i, define the (signed) net outflow of O_k induced by the directed obligation assignments in G by

$$\Delta_i^k(G) \;:=\; \sum_{(i\to j)\in G} q_{i\to j}^k \;-\; \sum_{(j\to i)\in G} q_{j\to i}^k,$$

where $q_{a\to b}^k$ is the quantity of O_k sent from a to b in the relevant directed obligation assignment. By Step 4, the MCR obligations for node i in traded object O_k are determined solely by $\Delta_i^k(G)$ (equivalently, i sends $(\Delta_i^k(G))_+$ and receives $(-\Delta_i^k(G))_+$).

Suppose Step 7 (Deficiency Removal) removes from G a collection of directed obligation-assignment portions (possibly via scaling factors $\alpha \in [0, 1]$) and let R denote the set of removed portions. Let the residual coordination group be

$$G' \;:=\; G \setminus R,$$

and let the MO re-net G' in Step 8 (canceling the prior MCR and generating a new MCR for G').

Consider any node i that is not a counterparty in any removed portion in R. Then every directed obligation assignment in R has counterparties (a, b) with $i \notin \{a, b\}$. Consequently, for each traded object O_k, the removed portions contribute zero to i's net outflow:

$$\sum_{(i\to j)\in R} q_{i\to j}^k \;=\; 0 \qquad \text{and} \qquad \sum_{(j\to i)\in R} q_{j\to i}^k \;=\; 0.$$

Moreover, the set (and quantities) of assignments in G that do involve i is unchanged when passing to G' (since no portion involving i was removed). Therefore, for every traded object O_k,

$$\Delta_i^k(G') \;=\; \Delta_i^k(G).$$

Since the netting step (Step 4) determines the amounts of each traded object that node i must send/receive from these net outflows alone, the re-netting from G to G' cannot change the traded objects sent or received by any node i who is not a counterparty to a removed directed obligation-assignment portion.

Finally, if a node is a counterparty in a removed portion, then at least one of the sums

defining $\Delta_i^k(\cdot)$ changes (for some k), so its net outflow (and thus its MCR send/receive amounts) may change. Hence, the only nodes that can experience a change in traded objects sent or received after re-netting are the counterparties in the removed directed obligation assignments. □

Corollary (Persistence of counterparty risk)**.** *When there is a re-netting after removing directed obligation assignments from a coordination group, the deficient node's original counterparty remains the counterparty for at least the amount of the deficiency.*

Proof. Fix a coordination group G in which a node A is deficient in some traded object O_k at Step 6. After applying the Rationing Rule (if relevant), let $\delta \geq 0$ denote A's remaining deficiency in O_k within G (i.e., the shortfall of committed O_k relative to A's required net outflow of O_k for the MCR associated with G).

By the Deficiency Removal Algorithm (Step 7), the MO allocates this deficiency across A's sending directed obligation assignments in G for traded object O_k by scaling those assignments: for each such assignment t (with A sending $q^k_{A\to b}(t)$ units of O_k to some original counterparty b), choose a scale factor $\alpha(t) \in [0,1]$ so that the removed portions $(1-\alpha(t))t$ satisfy

$$\sum_{t \in S^k_A(G)} (1-\alpha(t))\, q^k_{A\to b}(t) \;\geq\; \delta,$$

and (under the stated "remove only what is necessary" interpretation) the left-hand side equals δ. In words: the algorithm removes from A's sending assignments a total of δ units of the deficient traded object O_k.

Step 8 then (i) cancels the deficient MCR (retaining prior commitments) and (ii) issues bilateral contracts for each removed directed obligation-assignment portion. Each removed portion $(1-\alpha(t))t$ is, by definition, a portion of an initial bilateral contract between A and its original counterparty b, so the bilateral contract issued for $(1-\alpha(t))t$ has counterparties (A, b).

Therefore, for an aggregate amount of at least δ units of O_k (indeed exactly δ under exact removal), the obligations that cannot be funded inside the MCR are reinstated as bilateral obligations between A and the same original counterparties. Hence the counterparty (or set of counterparties) bearing the default exposure to A remains among A's original counterparties for at least the deficiency amount. □

Proposition 2. (Maximal multilateral netting of committed traded objects) *The NetWrap protocol achieves maximal multilateral netting of committed traded objects.*

Proof. Fix one execution of the protocol. Consider the set of residual coordination groups at the moment the protocol exits the inner loop (Step 8), so that every MCR induced by those groups is successfully committed under the COMMIT–VERIFY ALGORITHM (Step 6).

Fix any such residual coordination group G and any traded object O_k (in the exposition of the protocol, $O_k \in \{\text{y,x}\}$). Each directed obligation assignment in G is a directed traded object flow of the form

$$A \to B: \; q^k_{A\to B},$$

for some counterparties (A, B) and quantity $q^k_{A\to B} \geq 0$ (possibly zero for traded objects not exchanged in that assignment).

For each node i, define its net position in traded object O_k within G by

$$\Delta^k_i(G) \;:=\; \sum_{(j\to i)\in G} q^k_{j\to i} \;-\; \sum_{(i\to j)\in G} q^k_{i\to j}.$$

(Thus $\Delta^k_i(G) > 0$ is a net inflow and $\Delta^k_i(G) < 0$ is a net outflow.) Flow conservation implies

$$\sum_i \Delta^k_i(G) = 0 \qquad \text{for each } (G, k),$$

since every unit sent by some node is received by another node in the same group.

By the NETTING ALGORITHM (Step 4), the MCR for group G replaces the collection of assigned bilateral flows by each node's net obligations in G. Equivalently, for each node i and traded object O_k, the MCR specifies an

MCR net inflow: $\text{In}^k_i(G) := \big(\Delta^k_i(G)\big)_+$ and an *MCR net outflow*: $\text{Out}^k_i(G) := \big(-\Delta^k_i(G)\big)_+$,

where $(x)_+ := \max\{x, 0\}$. Hence, for each (G, k),

$$\sum_i \text{In}^k_i(G) \;=\; \sum_i \text{Out}^k_i(G),$$

and this common value is the gross quantity of O_k that must be transferred to implement the net positions $\{\Delta^k_i(G)\}_i$.

Now consider any feasible settlement procedure (not necessarily NetWrap) that clears the same net positions $\{\Delta^k_i(G)\}_i$ for traded object O_k in group G. Every node with $\Delta^k_i(G) > 0$ must receive at least $\Delta^k_i(G)$ units of O_k; therefore, any feasible procedure must transfer at least

$$\sum_{i:\,\Delta^k_i(G)>0} \Delta^k_i(G) \;=\; \sum_i \big(\Delta^k_i(G)\big)_+ \;=\; \sum_i \text{In}^k_i(G)$$

units of O_k in total within group G. NetWrap (via the MCR) achieves exactly this lower bound, and therefore minimizes gross transfers of O_k subject to the same net positions—i.e., it achieves maximal multilateral netting within G for that traded object.

In the multiple coordination group case, the protocol defines each node's aggregate MCR net outflow/inflow by summing MCR net outflows/inflows across coordination groups (NETTING ALGORITHM, multiple-group case). Step 6 commits these aggregate net outflows; Step 9 transfers traded objects in the amounts of net inflows. Since the argument above holds group-by-group and traded object-by-traded object, summing across groups preserves minimal gross transfer for each traded object given the induced aggregate net positions. Hence NetWrap achieves maximal multilateral netting of all traded objects that are actually committed (i.e., those remaining after the deficiency-removal inner loop terminates). □

Proposition 3. (Contractual profit invariance and counterparty-risk preservation) *Under the NetWrap protocol:*

(1) A node's contractual profit (i.e., profit if contractual obligations are fulfilled) is the same as under its initial directed obligation contracts.

(2) A node's counterparty risk remains with its initial directed obligation contract counterparties (i.e., the node is not exposed to default in a contract it did not initially enter into).

Proof. We use the protocol's terminology: initial directed obligation contracts (Step 1), directed obligation assignments and coordination groups (DIRECTED OBLIGATION ASSIGNMENT ALGORITHM, Step 3), MCRs (NETTING ALGORITHM, Step 4), commitment (COMMIT–VERIFY ALGORITHM, Step 6), and deficiency removal plus bilateral recovery (DEFICIENCY REMOVAL ALGORITHM, Step 7 and Step 8).

(1) Contractual profit is unchanged. By the DIRECTED OBLIGATION ASSIGNMENT ALGORITHM's Assignment Constraints (in particular constraint (ii)), for each initial directed obligation contract between counterparties (A, B) and for each traded object O_k in that contract, the sum of the quantities of O_k assigned to coordination groups equals the contract's original O_k flow between A and B; any unassigned remainder is returned/released in accordance with constraint (iii). Thus, relative to the initial directed obligation contract set, the protocol only (i) partitions each initial directed obligation contract into assigned portions (processed inside coordination groups) and possibly unassigned portions (not processed), without changing total contractual quantities.

Next, within each coordination group, the NETTING ALGORITHM replaces the assigned bilateral obligations by a multiparty coordination record (MCR) that specifies each node's net obligations per traded object within that group. Netting therefore preserves each node's net inflow/outflow induced by the assigned portions of its initial directed obligation contracts.

Finally, if a node fails to commit its full required (aggregate) MCR net outflow of some traded object in Step 6, then Step 7 applies deficiency removal and Step 8 issues recovered bilateral contracts for the removed directed obligation assignments. Each removed directed obligation assignment is (by definition) an initial directed obligation (or a portion thereof), so the recovered bilateral contracts preserve the original counterparties and the original traded object-flow terms on that removed portion.

Therefore, conditional on full performance of (i) the committed MCR obligations (which reproduce the net effect of the assigned portions) and (ii) any recovered bilateral obligations (which reproduce the removed portions), every node receives and delivers exactly the same total contractual traded object quantities as under its initial directed obligation contracts. Since contractual profit is computed from those contractual inflows/outflows at the original contractual terms, it is invariant under NetWrap.

(2) Counterparty risk remains with initial counterparties. Fix a node i. Consider any portion of i's initial-contract obligations during the protocol. There are two cases.

Committed (MCR-settled) portions. For any residual coordination group that survives the inner loop, Step 6 requires each node to commit (lock) the full amount of its aggregate MCR net outflow of each traded object prior to settlement. Once committed, the relevant traded objects are placed under protocol control (financial-institution lock or smart-contract escrow, COMMIT–VERIFY ALGORITHM), and Step 9 transfers traded objects according to net inflows. Thus, for committed portions, delivery to i is funded by traded objects already committed under the protocol; i does not become exposed to the default of any new node created by netting, and indeed post-commitment counterparty nonperformance cannot prevent delivery of already-committed traded objects.

Uncommitted (removed/recovered) portions. Any portion that is not successfully committed under Step 6 is removed in Step 7 and re-expressed as recovered bilateral contracts in Step 8. Because each recovered bilateral contract corresponds to an initial directed obligation assignment (i.e., an initial directed obligation contract or a portion thereof), its counterparties are exactly the counterparties from the initial directed obligation contract set. Hence any residual performance/default exposure attached to uncommitted portions remains with the initial counterparties.

In neither case does i acquire contractual exposure to a node with whom it did not initially contract. Therefore i's counterparty risk remains with its initial directed obligation contract counterparties. □

Proposition 3A. (Adjacency invariance) *Let $G_0 = (V, E_0)$ denote the exposure graph formed by the set of initial contracts, and let E_t be the edge set after any iteration t of the Deficiency-Removal Algorithm. Then $E_t = E_0$ for all t.*

Proof. Each removed directed obligation portion is reinstated as a bilateral contract between the same counterparties; no new edges are introduced and no edge endpoints are relabelled. □

Proposition 4. (Deterministic finite termination and computational bound) *For any execution of the NetWrap protocol (Algorithms 1–9) on a digital computing system with a finite number of nodes N and indivisible traded object units, the protocol terminates after at most V inner-loop iterations and within a bounded wall-clock time*

$$T = \mathcal{O}(V\,\tau \log N),$$

Here $V := \Phi_0$ denotes the initial total gross quantity of indivisible traded object units across all directed obligation assignments in the initial directed obligation set (equivalently, the initial value of the termination potential in Lemma 1), and τ is the mean network round-trip latency. Each iteration appends a verifiable, cryptographically signed transition record to the system transcript $\mathcal{L}$.

Lemma 1 (Mathematical Termination)**.** *Let Φ_t denote the total gross quantity of traded object units remaining in all residual coordination groups at iteration t. Then $\Phi_{t+1} < \Phi_t$ for every non-terminal iteration, and $\Phi_t \geq 0$ for all t. Consequently, after at most $\Phi_0 \leq V$ iterations the protocol halts.*

Proof. Each invocation of the DeficiencyRemove algorithm removes at least one positive quantity of some traded object k from the residual directed obligation set and reinstates it as a bilateral contract outside the iteration. Formally, if $\Phi_t = \sum_k \sum_{(A \to B) \in \mathcal{T}_t} q_k^{A \to B}$, then every inner-loop execution yields $\Phi_{t+1} \leq \Phi_t - 1$. Because Φ_t is a non-negative integer and strictly decreases whenever a deficiency occurs, there can be no infinite sequence of iterations. Hence the inner loop terminates after at most $V = \Phi_0$ iterations. □

Lemma 2 (Computational Realization)**.** *Assume the protocol executes on a network of N authenticated processor nodes implementing Algorithms 1–9, each node operating within bounded latency τ and using conflict-free replicated data types (CRDTs) for concurrent message ordering. Then each iteration performs at most $\mathcal{O}(N \log N)$ authenticated messages and $\mathcal{O}(V)$ arithmetic operations, implying total wall-clock time $T = \mathcal{O}(V\,\tau \log N)$.*

Proof. From Lemma 1, at most V iterations occur. Within an iteration, every node communicates its commitments and deficiency reports to at most $\log N$ peers through the CRDT dissemination tree, producing $\mathcal{O}(N \log N)$ total messages. Each message and local computation (commitment verification, deficiency scaling, and ledger update) completes within one network round trip τ. Hence cumulative communication delay is bounded by $\mathcal{O}(V\,\tau \log N)$. Because every message and state transition is signed and appended to the event log $\mathcal{L}$, the execution trace is deterministic and replay-verifiable without re-execution of settlement logic. □

Corollary (Deterministic machine termination)**.** *The event log $\mathcal{L}$ produced by a full NetWrap run contains exactly V signed state-transition records $\langle \text{state}_t, \text{op}_t, \sigma_{\text{MO}} \rangle$. Verification of termination therefore reduces to a polynomial-time scan of $\mathcal{L}$ and does not require re-running the settlement algorithms.*

Proof. By Lemmas 1 and 2, every iteration both (i) reduces Φ by at least one unit and (ii) emits a signed transition record. Because there are at most V reductions, exactly V records appear in $\mathcal{L}$. Checking signatures and the monotone decrease of Φ along the log is a linear-time procedure in $|\mathcal{L}|$, establishing deterministic and verifiable machine termination. □

E Implementation Variants and Design Extensions

This appendix collects optional implementation variants that extend deployability of NetWrap while preserving the formal control structure in Section 5. The variants below instantiate the protocol interfaces (Lock, Ration, Peel, Settle, Audit) and add operational controls for privacy, resilience, and anti-griefing without changing Steps 1–9. They are informative and do not alter required protocol behavior.

E.1 Privacy-preserving execution (TEE, MPC, and ZK proof interfaces)

In confidentiality-sensitive deployments, correctness is verified without broad disclosure of directed obligations or inventories. In a TEE implementation, the operator executes Steps 3–8 inside a remotely attested enclave and outputs (i) lock/commitment receipts bound to the protocol instance, (ii) signed MCR versions and cancellation events, and (iii) a hash-chained

transcript enabling integrity checks and replay. In an MPC implementation, nodes compute assignment and netting outputs on encrypted inputs, revealing only what is necessary to effect locking and settlement. In a ZK implementation, nodes (or the operator) produce succinct proofs for predicates such as: (a) allocated locks cover required outflows for a group; (b) peeled quantities equal recorded deficiencies; and (c) per-object conservation holds for each settled group.

The public inputs to a zero-knowledge audit may include the run_id, coordination-group identifier, terminal MCR hash, and transcript digest; the witness comprises the underlying transfer quantities, retained scale factors, and node identifiers. This allows an auditor to verify that peeled bilateral leftovers match removed quantities and that completion-phase transfers equal terminal net inflows without learning the full obligation set.

E.2 Distributed operator and governance controls

In a consortium implementation, multiple operator nodes jointly maintain protocol state and jointly authorize transitions. Authorization may use threshold signatures on transition records or BFT consensus to order events and reject invalid transitions. The transcript is replicated and cross-signed, enabling any node to verify authorization and hash continuity, and enabling recovery by replay from the last checkpoint without reconstructing directed obligations.

E.3 Optimization variants for deficiency removal and group scheduling

NetWrap admits multiple deterministic Peel policies subject to object-conservation and exposure-preservation constraints. Simple policies peel deficiency-linked portions by fixed priority orderings; optimized policies batch multiple deficiencies, prioritize actions that unblock the most groups, or select a maximal-feasible committed subset of residual assignments before peeling the remainder. Groups may be executed in parallel when independent, with deterministic transcript ordering to preserve replay and audit.

E.4 Anti-griefing controls: participation bonds, timeouts, and penalties

Some deployments deter strategic non-commitment by requiring a participation bond (or other stake) at enrollment or run start. If a node fails to commit required outflows within the commitment window, the protocol may apply a predefined penalty (e.g., forfeiture of all or part of the bond) and record the event in the transcript. Operational timeouts and exclusion rules can bound disruption (e.g., temporarily excluding repeat non-committers from subsequent runs) while preserving the underlying commit–verify–peel settlement semantics.

E.5 Backstop liquidity and exceptional cures (optional facility actions)

In settlement-window-critical markets, an optional backstop facility (e.g., regulator or designated liquidity provider) may cure a deficiency by supplying the deficient traded object to escrow, permitting settlement of an otherwise blocked netting group. Cure events are recorded as distinct transcript entries (amount, beneficiary, identifiers) for audit. This facility is orthogonal to the core protocol: deficiency removal remains the default mechanism; backstop actions are exceptional and enabled only under agreed governance rules.

E.6 Multi-asset, cross-rail, and cross-chain extensions

NetWrap generalizes to multiple objects and heterogeneous custody rails by treating Lock and Settle as adapters to external ledgers. Each traded object may have a distinct verification and settlement path, while the protocol maintains a unified transcript and applies deficiency removal object-by-object. For cross-rail deployments, additional safeguards (e.g., time-bounded locks, staged release ordering, or conditional releases) may be used to align finality across rails while preserving the requirement that no group settles unless required outflows are verifiably locked.