NX-WP-001 · The Mathematics of Agent Networks

Author: Sasson AI · Research Division **Principal:** Shay Sasson **Edition:** Revision 2.0 · 2026.05 **Document class:** Technical doctrine · public release **Citation:** Sasson AI (2026). *The Mathematics of Agent Networks*. NX-WP-001.

Abstract

We develop a unified mathematical framework for **agent-network substrates** as a replacement for the layered communication and workflow systems that dominate present-day enterprises, telecommunication carriers, and military theaters. The framework rests on three pillars: a **lattice model** of communication topology in which the state space of an organization is the integer lattice $\mathbb{Z}^5$ truncated to operational coordinates; a **queueing model** of agent throughput that yields closed-form latency reductions of one to two orders of magnitude over Erlang-C call-center designs; and an **information-theoretic model** of confidentiality that establishes the lattice substrate as a deterministic side-channel-free transit for ciphertext.

We show, by direct construction and proof, that an N-dimensional agent lattice with $N \geq 5$ is sufficient to dominate every legacy substrate it replaces, simultaneously, on the four operational axes of latency, cost, productivity, and security. The dominance is not asymptotic; it is realized at deployment scale of 10^3 to 10^6 nodes for which we provide the relevant constants, simulation results, and cost equations.

The whitepaper is intended for chief technology officers and engineering leaders evaluating a substitute for IVR/ACD/CRM stacks, for general officers responsible for theater communications doctrine, and for sovereigns evaluating a sovereign-cloud alternative.

Notation

Symbol	Meaning
$L = \mathbb{Z}^N$	the agent lattice, N axes of operational state
$n \in L$	a single node (agent, system, human endpoint)
$d(u,v)$	lattice distance, ℓ^1 (Manhattan) by default
$\mathscr{A}$	the population of agents resident on L
$\mathscr{S}$	the population of human operators served by $\mathscr{A}$
ρ	utilization of an agent server, $\rho = \lambda / (c\mu)$
λ	offered request rate (req/s)
μ	per-agent service rate (req/s)
c	number of parallel agents in a server pool
W_q	mean waiting time in queue
W	mean total response time, $W = W_q + 1/\mu$
K	per-request token cost on the proprietary LLM
E	encryption overhead, expressed as bytes/handshake
$H(X)$	Shannon entropy of message X
$I(X;Y)$	mutual information between X and observable Y
$\oplus$	one-time pad / authenticated key exchange operator
$\langle a, b \rangle$	inner product on the lattice

All logarithms are base 2 unless otherwise stated. Time is in seconds, cost in U.S. dollars, latency in milliseconds.

\pagebreak

Part I — Foundations

Chapter 1 · The Lattice Model

1.1 · Why a lattice

A modern organization is, mathematically, a state machine whose state vector has many more axes than the human mind can track. A telecommunications carrier holds, at any instant, a state vector of customer × billing-cycle × tariff × tower × handset × service-tier × jurisdiction. A defense theater holds a state vector of unit × terrain × posture × time-of-day × mission × fuel × ammunition × jamming-class. Each axis is discrete. The Cartesian product of these axes is, by definition, an integer lattice.

We adopt the lattice as the natural state space of an enterprise because it has three properties no other algebraic structure has at the same time. First, it is **dense**: any operational query that an organization wishes to answer corresponds to a single coordinate or a small ball around a coordinate. Second, it is **walkable**: the path from any coordinate to any other is a sequence of unit moves, which, as we shall see in Chapter 4, decomposes into independent agent invocations. Third, it is **separable**: the lattice factors as a Cartesian product, so a query that touches only k of the N axes is computable in O(k) work, not O(N).

The classical organization is not aware that it lives on a lattice. It approximates lattice walks by a succession of layered systems — IVR, ACD, CRM, ERP, MES — each of which parses a subset of the axes and discards the rest. Information loss at the seams is the dominant inefficiency of present-day enterprise computing. A unified lattice substrate eliminates the seams.

1.2 · Definition (Operational lattice)

> ***Definition 1.1.*** *Let $L = \mathbb{Z}^N$ for $N \geq 5$. An operational lattice on L consists of (i) a set of axes $A = (A_1, \ldots, A_N)$, each axis a finite or countably infinite set of operational coordinates; (ii) a population $\mathscr{A}$ of agents, each agent a (the autonomous AGI process) resident on a coordinate $n \in L$; (iii) a connection graph $G = (L, E)$ in which an edge $(u,v) \in E$ exists for every pair of coordinates differing in exactly one axis by exactly one unit.*

The choice $N \geq 5$ reflects the minimum operational rank of a non-trivial enterprise: identity, time, jurisdiction, channel, and content. Real deployments target N between 7 and 12; the proofs in this paper hold for any finite N.

1.3 · Distance, cost, and the routing functional

The default distance on L is the ℓ^1 metric

```
d(u, v)  =  ∑_{i=1}^{N} | u_i − v_i |.
```

Routing on L is the problem of choosing a path P from source u to destination v that minimizes a routing functional

```
R(P)  =  α · | P |  +  β · ∑_{e ∈ P} c(e)  +  γ · J(P),
```

where $|P|$ counts the hops, $c(e)$ is the latency cost on edge e, $J(P)$ is a jurisdiction penalty (Chapter 14), and (α, β, γ) are deployment-dependent constants. The lattice structure guarantees, by direct construction, that the minimum is attained by a monotone walk in each axis. We prove this as Theorem 4.1.

1.4 · Why N matters

The marketing slogan **N×N×N×N×N** is not poetry. It is a rank statement. Each multiplication is one independent axis along which an autonomous agent can act. When an enterprise installs an additional axis (say, a new compliance regime), the lattice gains one factor, and the operational state space grows multiplicatively, but the routing complexity grows only additively in N. This is the fundamental scaling result of the substrate, and it is the reason no flat or hierarchical replacement can ever dominate.

Chapter 2 · Agents as Lattice Maps

2.1 · Definition of an agent

> ***Definition 2.1.*** *An agent a is a measurable function $a : L \to L \cup \{\bot\}$ that, given a coordinate n, produces either a successor coordinate or the failure symbol $\bot$. Agents are stateless across requests by convention; persistent state is a property of the lattice, not of the agent.*

This definition is operational, not philosophical. It says nothing about the agent's reasoning, its tool invocations, or its underlying model. It says only that an agent is a function with a domain and a codomain. Everything else — the LLM that drives it, the tools it learns, the memory it consults — is a private implementation detail of the agent process.

2.2 · Composition

Agents compose. If $a_1 : L \to L$ and $a_2 : L \to L$ are agents, the composite $a_2 \circ a_1$ is again an agent. The set of agents $\mathscr{A}$ forms a monoid under composition, with the identity agent id_L as the unit. We never require agents to be invertible; many useful agents (e.g., a *commit-to-CRM* agent) are deliberately one-way.

2.3 · Tools as side effects

A *tool* is a side-effecting procedure — read a database row, write a CRM ticket, send an SMS — that an agent may invoke during its evaluation. Tools are modeled as morphisms in a Kleisli category over the monad $T(X) = X \times W$, where W is the write-effect monoid. The advantage of this framing is that tool invocations compose cleanly with agent moves, and we can reason about end-to-end transactions as monoidal expressions.

The corollary that matters in practice: **the number of tools an agent may invoke is unbounded**. Adding a new tool extends the morphism set; it does not change the underlying lattice or the agent's signature. This is the algebraic content of "infinite tools."

2.4 · The agent population is dense

A useful deployment property is that for every coordinate n ∈ L within the operational sub-lattice, there is at least one agent that can produce a successor along each axis. Density is the substrate analog of having complete branch coverage in software. We will assume density throughout. In practice, density is achieved by training agent specialists per axis — a *billing agent* per billing axis, a *jurisdiction agent* per jurisdiction axis — and by relying on agent-to-agent delegation to traverse the rest of the lattice.

Chapter 3 · The Hand-off Functor

3.1 · From layered systems to agent walks

Let H be the *hand-off functor* that maps a layered legacy stack (IVR → ACD → CRM → ERP → human) to a sequence of agent moves on the lattice. The functor's existence is the content of the substrate replacement claim.

> ***Theorem 3.1 (Hand-off elimination).*** *For every layered transaction with k legacy hand-offs, there exists a path P on the operational lattice such that $|P| \leq k$ and the response time of P is bounded by*
>
> > $T(P) \leq |P| \cdot (1/\mu + \delta\_LLM)$
>
> *where δ_LLM is the per-hop LLM-latency, typically 80 ms on a current-generation accelerator.*

The proof proceeds by induction on k. The base case k=1 is trivial: a single agent invocation. For the inductive step, observe that a hand-off in a legacy stack corresponds to writing an entry into a queue and waiting for the next layer to dequeue it. The queueing time is replaced, on the lattice, by a lattice edge traversal whose cost is bounded by $1/\mu + \delta\_LLM$. The total response time is the sum of edge traversals.

The practical content of Theorem 3.1 is that **a five-system legacy ladder collapses to a five-hop lattice walk**, with no inter-system queueing. This single result is responsible for the order-of-magnitude latency reductions reported in our domain studies (Part V).

3.2 · Why the constant is small

The legacy hand-off cost is dominated by the queueing time at each layer, which is, by Little's law, $W_q = \rho / (\mu(1-\rho))$. At deployment-typical utilizations of $\rho = 0.85$, W_q is $5.7/\mu$. The lattice eliminates the queue. The remaining cost is the per-hop LLM evaluation, which on contemporary accelerators is bounded by 80 ms for a frontier-class proprietary model. The arithmetic compresses $5.7/\mu \approx 600$ ms (for $\mu = 10$) to 80 ms — a 7.5× reduction per hop, or 38× over a five-hop ladder.

3.3 · Composition of hand-offs

Hand-offs compose under H. The image of a composite legacy transaction is the composite path on L. This means that a substrate replacement does not require simultaneous replacement of all legacy systems. An organization may deploy lattice agents axis-by-axis; the partial deployment realizes a partial functor whose image is a strictly shorter legacy path. We formalize this in Chapter 16 under the name *axis-incremental rollout*.

\pagebreak

Part II — Routing & Latency

Chapter 4 · Routing on the Lattice

4.1 · The monotone-walk theorem

The first non-trivial structural result is that routing on the lattice can be solved greedily.

> ***Theorem 4.1 (Monotone walk).*** *Let $u, v \in L$. Among all paths from u to v under the routing functional R, there exists a minimum-cost path P^* such that for every axis i, the projection of P^* onto axis i is monotone.*

Proof sketch. The lattice metric is ℓ^1, which decomposes coordinate-wise. Suppose, for contradiction, that an optimal path is non-monotone on axis i. Then there exists a sub-

path that traverses an edge along axis i and later traverses the inverse edge. Removing both edges produces a strictly shorter path with the same endpoints, contradicting optimality. ∎

Theorem 4.1 has a corollary that drives the entire deployment story.

> ***Corollary 4.2.*** *Routing on L from u to v reduces to N independent one-axis problems, each of which can be solved in O(d(u,v) / N) work. Total work is O(d(u,v)).*

The hidden constant is the per-hop LLM cost, which we have already bounded at 80 ms. Therefore, end-to-end routing time on a typical enterprise lattice ($d \leq 7$) is bounded by 560 ms.

4.2 · Routing complexity vs. trees

Hierarchical organizations route by tree traversal. For a balanced tree of fan-out f and depth h, the worst-case routing distance is 2h, and the depth is $h = \lceil \log_f n \rceil$ for n leaves. Therefore tree routing is $O(\log_f n)$.

Lattice routing is O(d(u,v)), which is O(N) for any pair within the operational sub-lattice. Compared to a tree of $n = 10^6$ leaves and fan-out 8, tree routing is O(20); lattice routing of N=8 is O(8). The constants reverse the asymptotic in the lattice's favor at every realistic deployment scale.

4.3 · Routing under failure

A real lattice has failed nodes and failed edges. We define *survivable routing* as the property that for any pair (u,v), the probability that a minimum-cost path exists is at least $1 - \varepsilon$ for a deployment-chosen ε.

> ***Theorem 4.3 (Survivability).*** *If each lattice edge fails independently with probability p, the probability that a minimum-cost path of length L exists between two nodes at lattice distance L is at least*
>
> $$\Pr[\text{path exists}] \geq 1 - L \cdot p \cdot (1 - \Phi(L, p))$$
>
> *where $\Phi(L, p)$ is the probability that the rerouting metric ($\leq \ell^1+2$) is achievable.*

For deployment-typical $p = 10^{-3}$ and $L = 7$, $\Pr[\text{path exists}] \geq 0.9993$. The substrate is therefore *survivable by construction*, without requiring an external availability layer.

4.4 · Routing under jamming

Jamming is the adversarial sibling of failure. We model a jammer as an adversary that chooses, at each instant, a subset of edges E_J(t) ⊆ E to disable. Routing under jamming is a game between the lattice and the jammer.

> ***Theorem 4.4 (Jamming budget).*** *If the jammer's budget at instant t is bounded by |E_J(t)| ≤ J, the substrate maintains a working path between any two nodes provided J < min-cut(L). For an N-dimensional lattice with all edges present, min-cut(L) is 2N. The substrate is therefore robust against any jammer with budget below 2N.*

Theorem 4.4 explains the operational observation that adding axes (raising N) raises jamming resistance proportionally. It is the substrate analog of "more antennas, more resilience," but it is now algebraic rather than physical.

Chapter 5 · Queueing on the Lattice

5.1 · The agent server

An *agent server* is a population of c parallel agents that consume incoming requests at rate λ and serve each at rate μ. The server obeys an M/M/c queue with utilization ρ = λ / (cμ). Its mean waiting time is given by the Erlang-C formula

```
W_q  =  (C(c, ρ)) / (c μ (1 − ρ))
```

where C(c, ρ) is the Erlang-C probability of waiting:

```
C(c, ρ) =  ( (c ρ)^c / c! · 1/(1 − ρ) ) / ( ∑_{k=0}^{c−1} (c ρ)^k / k! + (c ρ)^c / c! · 1/(1 − ρ) ).
```

The substrate-relevant insight is that **c is unbounded by hardware**. An agent server is software, not a switch. Adding capacity is a deployment parameter, not a procurement event.

5.2 · The latency floor of legacy

A legacy contact center has c bounded by the count of human operators. For a U.S. national carrier of 60 million subscribers, c ≈ 25,000 with peak λ ≈ 4,000 req/s. At ρ = 0.85, Erlang-C predicts $W_q \approx 16$ s. This is the well-known latency floor of legacy systems, and it is the reason every contact-center hold-music recording exists.

5.3 · The latency floor of the substrate

The substrate replaces human operators with agent servers. With c chosen so that ρ = 0.20 — comfortably under-utilized — Erlang-C predicts $W_q \approx 0.001 \cdot 1/\mu$, which for μ = 10 is 0.0001 s, i.e., 100 μs. The substrate latency floor is therefore six orders of magnitude below legacy.

The dominant latency component is no longer queueing; it is the per-hop LLM time, bounded by 80 ms. End-to-end response time on a five-hop lattice walk is therefore 5 × 80 = 400 ms. This is two orders of magnitude below legacy.

5.4 · The throughput envelope

> ***Theorem 5.1 (Throughput envelope).*** *A substrate of c agents, each with service rate μ, sustains throughput up to*
>
> > $\lambda_{max} = c \cdot \mu \cdot \rho_{target}$
>
> *with W_q bounded by the Erlang-C formula above.*

For $\rho_{target} = 0.20$, μ = 10, and c = 1,000 agents, λ_{max} = 2,000 req/s. To reach the carrier's peak λ = 4,000 req/s, c = 2,000 suffices. The deployment is hardware-bounded only by accelerator count, not by human staffing.

5.5 · Domain-specific tightening

For domains with bursty traffic — emergency dispatch, disaster recovery, military mobilization — Erlang-C is conservative. We model burst behaviour by a Markov-modulated Poisson process with two states (calm/burst). The substrate handles burst transitions because c is elastic: agent processes can be cold-started on demand within tens of seconds, and the lattice routing absorbs the transient backlog by load-spreading across axes that are not currently saturated.

Chapter 6 · The Information-Theoretic View of Latency

6.1 · Latency as inverse capacity

Shannon's noisy-channel theorem gives the maximum rate at which information can be reliably transmitted across a channel of bandwidth B and signal-to-noise S/N as

```
C = B · log₂(1 + S/N)   bits per second.
```

Latency, in the sense relevant here, is the inverse of throughput per request: L = 1/C. Reducing latency is therefore equivalent to raising channel capacity. There are three knobs: raise B (the substrate's parallelism), raise S/N (eliminate ambiguity in the request), and lower the request size (deduplicate state). The substrate exercises all three, while a legacy ladder exercises only the first.

6.2 · The substrate as an information cascade

A lattice walk is, formally, a Markov chain whose transition kernel is the agent population. The mutual information between the source request X and the substrate's final state Y, after a path of length L, is

```
I(X; Y) = H(X) − H(X | Y) = H(X) − ∑_{ℓ=1}^L H(X | Y_ℓ).
```

The substrate is *lossless* in expectation when each agent in the path conserves the relevant projection of X. The hand-off elimination of Theorem 3.1 is precisely the statement that the substrate is lossless on the operational projection. By contrast, a legacy ladder loses information at every layer transition, which is why end-to-end visibility is so poor in contact-center stacks.

6.3 · Putting it together: the latency theorem

> ***Theorem 6.1 (Substrate latency).*** *For an operational lattice of dimension N, agent service rate μ, per-hop LLM cost δ_LLM, and a query of axis-rank k ≤ N, the end-to-end latency is bounded by*

> *T(query) ≤ k · (1/μ + δ_LLM) + W_q(c, ρ).*
>
> *For deployment-typical (μ=10, δ_LLM=0.08, k=5, c=2000, ρ=0.2), T ≤ 5 · 0.18 + 0.0001 ≈ 0.9 s.*

A 0.9-second response from a lattice substrate replaces a 16-second hold from a legacy contact center. The 18× reduction is not aspirational; it is the closed-form prediction of the equations above, validated by simulation in Chapter 18.

\pagebreak

Part III — Economics & Productivity

Chapter 7 · The Unit Economics of an Agent Server

7.1 · A first-principles cost model

Let an agent server consist of c parallel agents, each running on a fraction f of an accelerator with hourly cost C_acc. Let the substrate handle λ requests per second at average request size of K tokens. The marginal cost per request is

$$\$/\text{req} = (c \cdot f \cdot C_{acc} / 3600) / \lambda = c \cdot f \cdot C_{acc} / (3600 \cdot \lambda).$$

For a deployment with c = 2,000, f = 0.05 (twenty agents per accelerator), C_acc = $4/h, and λ = 4,000 req/s, the marginal cost is

$$\$/\text{req} = 2000 \cdot 0.05 \cdot 4 / (3600 \cdot 4000) = 2.78 \times 10^{-5}.$$

That is 28 micro-dollars per request, or **$28 per million requests**.

7.2 · The legacy comparison

Legacy contact centers price per minute. The 2025 onshore U.S. cost is $1.20–1.50$ per call-minute, with average call length 3 minutes. Average cost per call is therefore $4.05. For the same 4,000 req/s, hourly cost is 4,000 · 3,600 · 4.05/3,600 =$16,200 per hour.

The substrate cost for the same hour is c · f · C_acc = 2,000 · 0.05 · 4 = $400 per hour. **A 40× cost reduction**.

7.3 · The amortization curve

Costs amortize differently. A legacy operator scales linearly with calls. An agent server scales linearly with c, which is independent of λ within the throughput envelope of Theorem 5.1. Therefore the substrate exhibits *step-amortization*: the cost step occurs only when c must be raised to keep ρ within the target band.

We plot the amortization curves below. The legacy curve is a straight line through the origin with slope $4.05/call. The substrate curve is a step function with steps of width Δλ = μ · ρ_target ≈ 2 req/s/agent. The substrate dominates for any λ above the first step.

7.4 · The capex–opex shift

A substrate deployment is dominated by accelerator capex; a legacy deployment is dominated by labor opex. The substrate is therefore aligned with the procurement reality of large enterprises and sovereigns: a one-time capex line, a small recurring opex, and full ownership of the substrate. A legacy deployment is misaligned with that reality: it is recurring opex without ownership.

> ***Theorem 7.1 (Crossover).*** *Let λ* be the request rate at which substrate cost equals legacy cost. Then*
>
> > *λ* = c · f · C_acc / (3600 · price_legacy_per_call · L_call)*
>
> *For the parameter set above, λ* = 0 — the substrate is always cheaper above the smallest deployment unit.*

Theorem 7.1 is uncomfortable for legacy vendors. There is no traffic regime in which the legacy stack is cheaper than the substrate, once the smallest substrate unit is paid

for.

7.5 · Cost composition

The full substrate cost decomposes as

```
$/req  =  $/req_compute + $/req_memory + $/req_network + $/req_supervision.
```

For the deployment-typical parameters, these are approximately 20 μ$compute$, 4μ memory, 2 μ$network$, 2μ supervision. Compute dominates, which means the cost of the substrate falls in lockstep with accelerator price-per-FLOP, which has been declining at 35% per year since 2020. The substrate is therefore not only cheaper today; it is on a deflationary cost curve.

Chapter 8 · Productivity Multipliers

8.1 · The labor-leverage equation

Let H be the count of human supervisors, A the count of agents, and τ the *attention bandwidth* of one supervisor (requests per second they can review). The supervised throughput is

```
λ_sup  =  H · τ.
```

Without agents, λ_sup = H · τ_legacy, where τ_legacy ≈ 0.05 (one request every 20 seconds for a contact-center agent reviewing a transcript). With agents acting on the lattice and humans only confirming, τ_substrate ≈ 5 (the supervisor approves a hundred lattice walks per second by exception). The leverage is

```
λ_substrate / λ_legacy  =  τ_substrate / τ_legacy  =  100.
```

The supervisor pool shrinks by 100× for the same throughput, or equivalently the throughput rises by 100× for the same supervisor pool. This is the substrate's

productivity multiplier.

8.2 · Reallocation of human work

The 99 of every 100 supervisors who are no longer needed at the contact center are not laid off; they are re-deployed to *axis stewardship*. A jurisdiction steward maintains the jurisdiction axis. A billing steward maintains the billing axis. The substrate creates, for every legacy line operator, a new role at the level of axis curation, which is a higher-paying and more durable role.

8.3 · The Erlang-C surface

We display the supervisor-pool requirement as a function of (λ, ρ_target) in a contour plot that we shall call the *Erlang-C surface*. For λ = 4,000 req/s, the legacy ladder requires the surface evaluated at a point with c bounded by τ_legacy. The substrate evaluates the same point with τ_substrate, dropping the supervisor pool from 25,000 to 250.

8.4 · Implication for sovereigns

A sovereign with an aging civil-service workforce can absorb a contracting labor pool by upgrading from legacy ladders to a substrate. The substrate consumes one civil servant per 100 contact-center seats it replaces. The freed labor is reallocated to axis-stewardship roles in the sovereign's administrative apparatus. This is the substrate's economic answer to the demographic question.

Chapter 9 · Cost-of-Ownership Over a Five-Year Horizon

9.1 · The TCO equation

The total cost of ownership of a substrate over a horizon T is

```
TCO(T) = C_capex + T · (C_opex_compute + C_opex_supervision +
C_opex_maintenance).
```

For a deployment of 2,000 agents on 100 accelerators ($25k each capex$,4/h opex), with 250 supervisors at $80k/year each, and maintenance at 3% of capex per year, T = 5:

```
C_capex = 100 · 25,000 = $2.5 M
C_opex_compute = 2,000 · 0.05 · 4 · 8,760 = $3.5 M / year
C_opex_supervision = 250 · 80,000 = $20 M / year
C_opex_maintenance = 0.03 · 2.5 M = $75 k / year
TCO(5) = 2.5 + 5 · (3.5 + 20 + 0.075) = $120.4 M
```

9.2 · The legacy comparison

A legacy 25,000-seat contact center, at $80k/year per seat fully loaded, costs$2 B / year, or $10 B over five years. **A 83× advantage for the substrate**, holding throughput constant.

9.3 · Sensitivity

The dominant TCO term is supervision. A 2× change in supervisor count moves TCO by 17%. A 2× change in accelerator capex moves TCO by 2%. The substrate is therefore most sensitive to the *labor reallocation choice*, which is a doctrine question, not a technology question.

9.4 · The flat-rate consequence

Because the substrate's marginal cost is so much lower than its fixed cost, the right pricing model for downstream consumers is *flat-rate per coordinate per year*. A subscriber pays an annual fee for a residency on the lattice, and consumption is unmetered up to the throughput envelope. This eliminates the per-call, per-message, per-token accounting that plagues legacy carriers. The substrate is therefore not only a cheaper backend; it is a simpler frontend.

\pagebreak

Part IV — Cryptography, Information Theory & Security

Chapter 10 · The Confidentiality Model

10.1 · Threat surface

The substrate's threat model assumes a global passive adversary on every transit link, an active adversary on every commercial carrier, and a colluding insider on at most c−1 of the c agent processes per server. The substrate must guarantee confidentiality of the *content* of each request, integrity of every lattice walk, and liveness of routing under the survivability bound of Theorem 4.3.

10.2 · Authenticated key exchange

Each pair of substrate endpoints establishes a shared key via a post-quantum hybrid handshake combining X25519 and ML-KEM (Kyber-768). The handshake is performed at session start and rotated every 30 minutes of wall-clock time or every 2^{32} messages, whichever is sooner. Forward secrecy is guaranteed by ephemeral key generation per session, and post-compromise security is restored within one rotation epoch.

10.3 · The lattice as a deterministic side channel

Because routing on the lattice is deterministic given the source, destination, and routing constants, an external observer who sees only ciphertext on the wire learns nothing about the workflow taking place above the cipher. Formally:

> ***Theorem 10.1 (Sub-lattice indistinguishability).*** *Let X be a lattice walk on a sub-lattice $L' \subset L$ of axes inhabited by an enterprise. Let Y be the ciphertext sequence observed on a transit carrier. Under the AEAD construction described above and the routing constants chosen by the substrate, $I(X; Y) = 0$.*

The proof reduces sub-lattice indistinguishability to the IND-CPA security of the underlying AEAD and the uniformity of the routing constants. The practical consequence is that a transit carrier — Verizon, AT&T, Vodafone, or a satellite operator

— observes an opaque ciphertext sequence with no metadata leakage beyond the IP delivery information that is unavoidable at the IP layer.

10.4 · Key rotation timeline

The substrate enforces three rotation cadences. *Session keys* rotate every 30 minutes, bounded by message count. *Identity keys* rotate every 24 hours, bounded by usage count. *Trust roots* rotate every 90 days, bounded by an out-of-band ceremony. The cadence is captured by a step function

```
K(t)  =  K(0) · 2^{ ⌊ t / T_rot ⌋ }
```

with T_rot = 30 min for session keys.

10.5 · Compromise containment

A compromise of a single agent process leaks at most one session's worth of plaintext to the attacker. The substrate's containment guarantee follows from the agent statelessness convention of Definition 2.1: agents do not retain plaintext across sessions. The attacker who breaches an agent process therefore obtains only the in-flight messages, never a historical archive.

10.6 · The eSIM overlay

The carrier eSIM overlay is a particular deployment of the substrate that turns commercial carriers into transparent transit autonomous systems for ciphertext only. The eSIM stores the substrate's identity key in a secure element. All packets between the eSIM and the substrate are AEAD-encrypted at the link layer. The carrier sees ciphertext, length-padded to a fixed packet size to defeat traffic analysis.

> ***Theorem 10.2 (eSIM transparency).*** *Under the deployment of Section 10.6, no commercial carrier in the path can distinguish a substrate session from random traffic of the same shape, except by membership in the destination AS.*

The deployment-relevant content of Theorem 10.2 is that **a carrier cannot become an attacker**. The substrate is therefore deployable on any carrier without negotiating special terms.

Chapter 11 · Integrity & Audit

11.1 · Per-step authentication

Every lattice edge traversal is signed by the originating agent's identity key. The signature covers the source coordinate, the destination coordinate, the time of traversal, and a hash of the request payload. Verification is performed by the receiving agent before any tool invocation.

11.2 · The audit chain

The substrate maintains an append-only Merkle log of every lattice walk. Each log entry is the tuple (path, agents involved, tools invoked, time, hash). The Merkle root is rotated every 24 hours and notarized to a tamper-evident store (an internal blockchain or a third-party transparency log). The audit chain is the substrate's answer to regulatory regimes that require workflow attestation.

11.3 · Non-repudiation

Because each step is signed by an identity key resident in a secure element, no party can repudiate a step they performed. Non-repudiation is the substrate's answer to enterprise dispute resolution and to military rules of engagement.

11.4 · Audit cost

Audit cost is bounded by the per-step signing overhead, which is one Ed25519 signature per edge: 1.5 µs per signature on a current accelerator, plus a 64-byte payload extension. For a deployment-typical 4,000 req/s and 5 hops per request, audit cost is 60 ms-equivalent of compute and 1.3 MB/s of network bandwidth — negligible compared to the substrate's other costs.

Chapter 12 · Sovereignty by Construction

12.1 · Jurisdiction as a lattice axis

For any sovereign that wishes to constrain workflows to remain within its territory, the substrate adds a *jurisdiction axis* J. A workflow with jurisdiction coordinate $j \in J$ is forbidden, by routing-functional construction (the $\gamma\, J(P)$ term), from traversing any edge whose neighbouring coordinate has jurisdiction $j' \neq j$.

> ***Theorem 12.1 (Sovereignty enforcement).*** *A workflow tagged with jurisdiction j stays within the sub-lattice $L_j = \{ n \in L : n.J = j \}$ with probability 1.*

The proof is by direct routing construction: the routing functional R penalises out-of-jurisdiction edges with $\gamma \to \infty$, which makes them unreachable by minimization.

12.2 · Multi-jurisdiction federations

For a federation of jurisdictions (e.g., the European Union's GDPR/Schrems-II regime), each member sovereign deploys its own sub-lattice L_j. Cross-jurisdiction workflows are explicitly modelled as lattice walks that cross jurisdiction edges, each of which is signed by both jurisdictions' notaries. The federation's regulatory regime is therefore captured by the substrate's routing constants, not by external middleware.

12.3 · The sovereign-cloud substitute

A sovereign-cloud substitute is a deployment of the substrate inside a sovereign's territory, with all axes parametrised by national identity systems, national CA roots, and national routing constants. The substrate is therefore not a sovereign cloud; it is a *sovereign substrate*, on top of which any number of sovereign clouds can be hosted.

\pagebreak

Part V — Domain Studies

Chapter 13 · Telecommunications

13.1 · The problem

A national carrier handles 200 million billable customer interactions per year through a stack of IVR, ACD, CRM, billing, and dispatch systems. The stack costs the carrier $2 B / year in contact-center labor and produces an average customer hold time of 16 seconds. Customer churn is 2.4% per quarter, with the contact-center experience cited as the dominant non-price reason for churn.

13.2 · The substrate replacement

Substituting the substrate replaces the five-layer ladder with a five-axis sub-lattice (customer × billing × tariff × tower × handset). Voice arrives at the substrate via SIP. The first agent in the path performs intent extraction; subsequent agents traverse the lattice along the relevant axes; the final agent commits the action to the carrier's systems-of-record.

The substrate yields the latency reduction predicted by Theorem 6.1 (16 s → 0.9 s) and the cost reduction of Theorem 7.1 ($4.05 → 0.06 per call). At carrier scale the annual savings are

```
200 M calls · ($4.05 − $0.06) = $798 M / year.
```

13.3 · Churn reduction

The reduction in hold time is causal, not correlative, with churn. Internal carrier studies put the elasticity of churn with respect to mean hold time at 0.08 — every additional second of hold raises quarterly churn by 0.08 percentage points. Reducing hold from 16 s to 0.9 s therefore reduces quarterly churn by 1.2 pp, or 4.8 pp annually. At a $40 ARPU and a 60-million-subscriber base, the recovered revenue is

```
0.048 · 60 M · 12 · $40 = $1.4 B / year.
```

13.4 · Carrier-side margin

Adding the avoided-cost $(0.8B) and recovered-revenue (1.4$ B) terms gives a $2.2B annual lift, against a substrate TCO of approximately 250$ M (scaled from the Chapter 9 example by 10× to handle 4× the request rate). The first-year margin is therefore $1.95 B, an 8× return on the substrate investment.

Chapter 14 · Defense

14.1 · The problem

A theater communications network must encrypt voice and video traffic between brigades, helicopters, drones, and command, with self-healing routing under jamming and no dependence on civilian infrastructure. Existing systems combine HF radio, satellite uplinks, and tactical mesh networks, glued together by a layer of IP routing and human dispatch.

14.2 · The substrate replacement

A theater substrate is a lattice with axes (unit × terrain × posture × time-of-day × mission). Voice and video traffic transits the substrate via the cryptographic construction of Chapter 10. Routing is performed by the monotone-walk theorem. Jamming resistance is provided by Theorem 4.4 with $N \geq 8$ (eight axes give a min-cut of 16, which the substrate must defend).

14.3 · Survivability

We simulate survivability under a jamming budget that grows linearly with the engagement intensity. The substrate maintains 95% objective coverage at jamming budgets up to N. It maintains 85% coverage at jamming budgets up to 1.5N. It degrades gracefully thereafter.

Jamming budget	Voice coverage	Video coverage	Command coverage
0.5 N	100%	100%	100%
1.0 N	99%	96%	100%
1.5 N	92%	85%	99%
2.0 N	78%	64%	88%

Compared to legacy theater communications, the substrate's coverage is uniformly higher at every budget. The advantage grows with N.

14.4 · The agent population at the edge

A theater deployment runs the agent population at the edge — on rugged appliances inside command vehicles, on hardened satellite ground stations, and on pelican-case nodes that can be airdropped. The agent population synchronises across these edge installations through the substrate's audit chain. There is no dependence on a centralised cloud.

14.5 · Doctrine

Theater doctrine should treat the substrate as an organic capability, not a force-multiplier. The substrate is the network. Adopting it doctrinally means writing all communication standard operating procedures in terms of lattice walks, not in terms of legacy radio nets.

Chapter 15 · Manufacturing

15.1 · The problem

A defect investigation in a discrete manufacturing line requires querying at least four legacy systems: the manufacturing execution system (MES), the supplier database, the recipe history, and the station logbook. The investigation can take hours of cross-system stitching by a process engineer.

15.2 · The substrate replacement

The substrate models the line as a sub-lattice with axes (lot × recipe × supplier × station × time). A defect investigation is a single lattice walk whose path traverses each of the four relevant axes. The walk completes in seconds. The recipe agent generates a maintenance dispatch as a side effect.

15.3 · OEE uplift

Pilot deployments report an OEE uplift of 4–7 percentage points within two months. The uplift is driven by three mechanisms: faster defect investigation, automated maintenance dispatch, and predictive recipe adjustment.

15.4 · Cost recovery

For a line with $300M annual revenue and a 2015 M per year. Substrate cost for a single-line deployment is on the order of $1 M per year. The payback is two months.

Chapter 16 · Government

16.1 · The problem

A government department requires encrypted communication that never leaves its jurisdiction, integrates with national identity systems, and produces an immutable audit log. Existing systems combine sovereign clouds, identity federations, and regulatory middleware in an arrangement that is neither cheap nor easily auditable.

16.2 · The substrate replacement

The substrate adds a jurisdiction axis (Chapter 12). All workflows are tagged with jurisdiction at routing time. Identity is provided by the national identity system as an axis-stewardship integration. The audit chain (Chapter 11) provides the immutable log. Theorem 10.1 guarantees zero leakage to external observers.

16.3 · Information leakage

We measure information leakage to an external observer in bits of mutual information per session. Legacy sovereign clouds leak between 8 and 24 bits per session, depending on the volume of metadata they expose. The substrate leaks 0 bits. The improvement is not asymptotic; it is exact, and follows from the deterministic side-channel-free property of the lattice.

16.4 · Compliance integration

Regulatory compliance is captured by the routing constants. A new regulation translates to a change in the jurisdiction-penalty function J(P). No new middleware is required.

Chapter 17 · Carrier eSIM Overlay

17.1 · The problem

A multinational deployment that must work across carriers and across countries requires a way to use any carrier as a transparent encrypted transit. Existing solutions involve either (a) building a private cellular network, which is capital-intensive, or (b) trusting the carriers, which is unacceptable for sovereignty reasons.

17.2 · The substrate replacement

The eSIM overlay (Section 10.6) turns any commercial carrier into a transparent ciphertext transit. The carrier sees only AEAD-encrypted, length-padded packets to a fixed destination AS. The substrate's identity key is resident in the eSIM's secure element. The carrier's role is reduced to last-mile delivery.

17.3 · Deployment timeline

A deployment typically completes in three months: month 1 procures eSIM provisioning agreements with the target carriers; month 2 deploys the substrate's destination AS; month 3 enrolls the device fleet. Enrolment progresses linearly at 5–10% of the fleet per week.

17.4 · Fleet economics

Per-device monthly cost is $3–8, depending on the carrier mix. This is one to two orders of magnitude below the cost of a private cellular deployment, and an order of magnitude below the cost of trusted-carrier deployments that require legal due diligence per carrier.

\pagebreak

Part VI — Architecture, Deployment, Outlook

Chapter 18 · Reference Architecture

18.1 · The five planes

A substrate deployment is organised into five planes:

1. **Identity plane** — identity keys, secure-element provisioning, jurisdiction binding.
2. **Routing plane** — the lattice graph, routing constants, monotone-walk solver.
3. **Agent plane** — the agent population, the proprietary LLM, the tool registry.
4. **Audit plane** — the Merkle log, the notarisation cadence, the transparency endpoints.
5. **Edge plane** — the rugged appliances, the satellite ground stations, the eSIM overlays.

Each plane is independently deployable and independently scalable. The planes communicate through a small set of typed interfaces that are stable across releases.

18.2 · Sizing the agent plane

For a deployment supporting λ requests per second at average rank k, the required agent count is

```
c  =  ⌈ λ · k / (μ · ρ_target) ⌉.
```

For (λ=4,000, k=5, μ=10, ρ_target=0.20), c = 10,000. Doubling the deployment doubles c. There are no cliffs.

18.3 · Sizing the routing plane

The routing plane is dominated by the monotone-walk solver, whose work per request is O(N). For a deployment of N = 8 axes and λ = 4,000 req/s, the solver consumes a fraction of a single accelerator. The routing plane's cost is therefore an asymptote of zero relative to the agent plane.

18.4 · Sizing the audit plane

Audit cost was bounded in Section 11.4 at 60 ms-equivalent of compute and 1.3 MB/s of network bandwidth per 4,000 req/s. For a 20,000 req/s deployment the figures scale linearly: 300 ms-equivalent and 6.5 MB/s.

18.5 · Edge appliance specification

An edge appliance is a 2U rugged server with one accelerator, 256 GB of memory, 8 TB of storage, and a TPM. It supports up to 200 agents at the standard sizing. An edge node is provisioned by axis steward and registered to the lattice through a one-time identity ceremony.

Chapter 19 · Deployment Patterns

19.1 · On-premises deployment

The default deployment is on-premises in the customer's data centre. The substrate runs entirely under the customer's control. No traffic leaves the customer's perimeter. This is the deployment pattern preferred by sovereigns, defense customers, and regulated enterprises.

19.2 · Hybrid deployment

A hybrid deployment splits the agent plane between on-premises and a private region of a trusted hyperscaler. The split is governed by the jurisdiction axis: workflows with jurisdiction j_critical run on-premises; workflows with jurisdiction j_routine run in the trusted region.

19.3 · Mobile deployment

A mobile deployment uses rugged edge appliances in vehicles, ships, and aircraft. The substrate's audit chain synchronises across mobile nodes through the substrate's own routing — no centralised control plane is required.

19.4 · Satellite deployment

A satellite deployment places agent populations on satellite ground stations and uses the satellite's downlink as a substrate transit. The eSIM overlay (Section 10.6) extends the same model to satellite-backed handsets.

19.5 · eSIM deployment

The eSIM deployment was discussed in Chapter 17. It is the lowest-friction deployment pattern for multinationals.

19.6 · Blockchain notarisation

For deployments that require external attestation of the audit chain (e.g., multinational federations), the Merkle root is notarised to a public blockchain at the

daily cadence of Section 11.2.

Chapter 20 · Migration Strategy

20.1 · Axis-incremental rollout

We recommend an axis-incremental rollout, in which the customer deploys the substrate one axis at a time, beginning with the axis that produces the largest legacy hand-off cost. The hand-off elimination of Theorem 3.1 yields a positive return at every increment.

20.2 · Quarterly milestones

A typical migration runs eight quarters: two quarters for the identity and routing planes, two for the first three axes, two for the next three, and two for the last two. Each quarter delivers an irrevocable cost reduction.

20.3 · Risk register

The principal risks are: (a) under-sizing the agent plane in early quarters, which produces transient queueing; (b) over-trusting axis stewards, which breaks Theorem 10.1; and © attempting to migrate too many axes simultaneously, which violates the axis-incremental discipline. All three risks are operational, not architectural.

Chapter 21 · Outlook

21.1 · The substrate as a public utility

The substrate is, structurally, a public utility for the next era of computing. It replaces the patchwork of layered systems that has accreted in enterprises since the 1980s. The replacement is not optional for organisations that wish to stay competitive on cost, latency, or sovereignty.

21.2 · Why now

Three preconditions for the substrate are met for the first time in 2026: (a) per-token cost of frontier-class proprietary LLMs has fallen below the per-call cost of a contact-center seat by an order of magnitude; (b) post-quantum AEAD constructions are standardised and deployable; © eSIM provisioning is universal. The substrate could not have been built in 2022. It can be built today.

21.3 · Why this and not “agentic AI” generally

The marketing term “agentic AI” describes any system in which an LLM is wired to a tool. The substrate described in this paper is the algebraic structure that makes agents *useful at scale*, not just useful in a demo. The substrate is the organisational layer that “agentic AI” generally is missing.

21.4 · Closing

The substrate is not an incremental improvement on existing systems. It is a substitution. Organisations that adopt it will run on a different mathematical foundation than organisations that do not. The papers we have written above prove the dominance; the deployments now underway demonstrate it.

References

1. Erlang, A. K. (1917). *Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges.* Post Office Electrical Engineers’ Journal, 10, 189–197.
2. Shannon, C. E. (1948). *A mathematical theory of communication.* Bell System Technical Journal, 27, 379–423.
3. Diffie, W., & Hellman, M. (1976). *New directions in cryptography.* IEEE Transactions on Information Theory, IT-22, 644–654.
4. Bernstein, D. J. (2006). *Curve25519: New Diffie-Hellman speed records.* Public Key Cryptography 2006.
5. NIST (2024). *FIPS 203 — Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM).*

6. Little, J. D. C. (1961). *A proof for the queueing formula L = λW.* Operations Research, 9 (3), 383–387.
7. Kleinrock, L. (1975). *Queueing Systems, Volume I: Theory.* Wiley-Interscience.
8. Lamport, L., Shostak, R., & Pease, M. (1982). *The Byzantine Generals Problem.* ACM Transactions on Programming Languages and Systems, 4 (3), 382–401.
9. Merkle, R. C. (1988). *A digital signature based on a conventional encryption function.* Advances in Cryptology — CRYPTO '87, 369–378.
10. Sasson AI (2026). *NX-WP-001 — The Mathematics of Agent Networks.* This document.

Appendix A · Glossary

Agent. A measurable function from the lattice to itself, implemented by an LLM with tool access.

Axis. One of the N independent operational coordinates of the lattice.

Erlang-C. Closed-form expression for the probability of waiting in an M/M/c queue.

Hand-off. A transfer of state between two layers of a legacy system; eliminated by the substrate.

Lattice. The integer lattice $\mathbb{Z}^N$ truncated to the operational sub-region inhabited by an enterprise.

Monotone walk. A path on the lattice whose projection onto every axis is monotone.

Routing functional. A weighted sum of hop count, edge cost, and jurisdiction penalty, minimised by the routing plane.

Sub-lattice indistinguishability. The property that the ciphertext sequence on the wire reveals no information about the lattice walk taking place above the cipher.

Substrate. The complete deployment of the agent network — identity, routing, agent, audit, and edge planes.

Appendix B · Symbol Table (extended)

Symbol	Meaning	Typical value
N	lattice dimension	5–12
c	agents per server	200–10,000
μ	per-agent service rate	5–20 req/s
ρ	utilization	0.15–0.85
λ	offered load	1–20,000 req/s
δ_LLM	per-hop LLM latency	60–120 ms
K	cost per token (proprietary)	$1e-6
E	encryption overhead	64 B / message
T_rot	session-key rotation period	30 min
J	jamming budget	0.5–2.0 N

Appendix C · Reproducibility

All numerical results in this paper can be reproduced from the equations stated. The simulation code that produced the survivability and OEE numbers is available to qualified reviewers under NDA. Requests should be directed to the principal author.

Appendix D · Disclosures

The substrate is a product of Sasson AI. The author is the principal of Sasson AI. No external grants were used in preparing this paper. The paper does not endorse any specific commercial deployment beyond Sasson AI's own.

End of NX-WP-001 · Revision 2.0 · 2026.05