I.F.P.P ANNEX A :

IFPP MESSAGE PACKET SPECIFICATION AND STATELESS TRANSPORT BEHAVIOR

OCTOBER 15, 2025

(IFPP WHITEPAPER ANNEX AND DEFINITION)

ENGR JUAN CARLOS G AYENG,
BACOLOD CITY, THE PHILIPPINES

DECENTRALIZED
AI-Assisted
Peer-to-Peer
Messaging Protocol
Using Fractal Propagation
with
LOW Footprint
&
Complete Privacy

I.F.P.P. PROTOCOL ANNEX

ANNEX A — IFPP MESSAGE PACKET SPECIFICATION AND STATELESS TRANSPORT BEHAVIOR

(“IFPP MESSAGE PACKET SPECIFICATION AND STATELESS TRANSPORT LAYER BEHAVIOR (NO-SOCKET, NO-SESSION TRANSPORTS)”)

A.1. OVERVIEW

The Intelligent Fractal Propagation Protocol (IFPP) redefines the transport behavior of digital messages across heterogeneous, self-forming swarms of devices.
Unlike TCP/IP, which depends on sockets, sessions, and acknowledgments, IFPP employs stateless, event-driven exchanges designed for autonomous relay environments.

Each IFPP message is composed of three interdependent yet independently storable components:

1. ***EPHEMERAL HEADER** — transient bootstrap and authentication data.*
2. ***AI METADATA** — dynamic, self-evolving contextual information for learning and traceability.*
3. ***ETERNAL MESSAGE CORE** — immutable encrypted payload representing the true message.*

All IFPP transmissions occur without sockets or streams, using one-shot, signed, self-destructing datagrams exchanged over any available carrier (UDP, Bluetooth, Wi-Fi, or TCP/IP bridge).

Every device acts as a temporary router, verifier, and teacher, ensuring network resilience and self-healing.

A.2. IFPP MESSAGE STRUCTURE

Component	Description	Storage Class	Example Size	Persistence	Contents
Ephemeral Header	*Transient routing + identity data*	*Volatile*	*1–2 KB*	*Milliseconds*	*Fractal Hop ID, Device UUID, Seed Hash, Angel Team Token*
AI Metadata	*Progressive swarm intelligence data*	*Semi-volatile*	*2–5 KB*	*Per-hop*	*Hop Digest, Traceback Map, Gabriel Extracts, Device UUID Pairings*
Eternal Message Core	*Immutable encrypted message*	*Persistent*	*10 KB–10 MB*	*Permanent*	*Encrypted Payload, Swarm Key Pair, Sender/Receiver IDs*

A.2.1. EPHEMERAL HEADER (SEED DATA)

- ***FRACTAL HOP ID (FHI)*** *– e.g., a1, a1b1, a1b2; universal fractal lineage of message path.*
- ***DEVICE UUID*** *– locally unique identifier cryptographically bound to hardware.*
- ***SEED HASH*** *– derived from swarm public + device private key, ensuring authenticity.*
- ***ANGEL TEAM TOKEN*** *– minimal executable code seed that re-instantiates the 7-Angel microkernel on the receiving node.*
- ***SOURCE/DESTINATION DIGEST*** *– compact hashes used for routing and visual validation.*

A.2.2. AI METADATA

- ***HOP DIGEST** – signed record of device UUIDs and fractal branch lineage.*
- ***PARTIAL LEARNING SNAPSHOT** – per-hop device AI state delta (not fully transmitted).*
- ***TRACEBACK INDEX** – reverse mapping of all encountered nodes for back-propagation.*
- ***FEDERATED LEARNING EXTRACT** – vector update from Gabriel's swarm learning module.*

Transmission Rule:

AI Metadata is locally appended, but only hashed digests are propagated.

The complete intelligence graph emerges collectively across the swarm — no node holds the full picture.

A.2.3. ETERNAL MESSAGE CORE

- ***ENCRYPTED PAYLOAD** – sealed using sender's private and receiver's public key.*
- ***CORE SIGNATURE** – immutable signature of source identity + Sacred Persistence flag.*
- ***SWARM KEY PAIR (SPK/SRK)** – ephemeral encryption keys for each propagation wave.*
- ***INTEGRITY ANCHORS** – Gabriel's checksum linking metadata lineage and hop chain.*

Decryption Rule:

Only the legitimate destination device can open the Eternal Core, determined by the Swarm Receiver Key (SRK).

Intermediary nodes forward or verify, but cannot decrypt.

A.3. MODE 1 — APP → UDP

(STATELESS DATAGRAM)

Flow:

APP (TEAM LEADER)

↓

ENCODE IFPP PACKET (EPHEMERAL + METADATA + CORE)

↓

LIAISON ANGEL SELECTS UDP INTERFACE

↓

CREATE ONE-SHOT DATAGRAM

↓

SENDTO(DEVICE_IP, RANDOM_PORT)

↓

POINT ANGEL VERIFIES TRUST HASH

↓

DESTROY SOCKET IMMEDIATELY

Behavior:

- No connect() or 3-way handshake.
- Random ephemeral port.
- Receiver spawns micro-listener (milliseconds), authenticates, self-destructs.

Analogy:

A signed flare tossed across the sky — visible long enough to verify, then gone.

A.4. MODE 2 — APP → BLUETOOTH

(LOW ENERGY / CLASSIC)

Flow:

APP (TEAM LEADER)

↓

LIAISON ANGEL SCANS NEARBY BLUETOOTH NODES

↓

RECON VERIFIES CANDIDATE IN GABRIEL REGISTRY

↓

POINT ANGEL FORMS SHORT-LIVED L2CAP/GATT CHANNEL

↓

TRANSMIT IFPP PACKET

↓

IMMEDIATE DISCONNECT

Behavior:

- No pairing or bonding required.
- Uses BLE advertisement or GATT write for envelope.
- Encryption entirely handled by IFPP layer.

Analogy:

A sealed note passed hand-to-hand — no lingering connection.

A.5. MODE 3 — APP → WI-FI

(AD-HOC OR MESH)

Flow:

APP (TEAM LEADER)
↓
LIAISON ANGEL JOINS WI-FI DIRECT OR MESH
↓
RECON SCANS FOR IFPP-BROADCAST DEVICES
↓
POINT ANGEL SENDS BROADCAST FRAME
↓
RECEIVER DECODES, VALIDATES UUID
↓
IF NOT DESTINATION → FORWARD FRACTALLY

Behavior:

- Operates outside or above IP.
- Stateless, broadcast-capable.
- Ideal for enclosed mesh or swarm networks.

Analogy:

A crowd passing coded notes — each node decides instantly to keep or forward.

A.6. MODE 4 — APP → TCP/IP BRIDGE

(LEGACY COMPATIBILITY)

Flow:

APP (TEAM LEADER)

↓

LIAISON ANGEL ROUTES TO IFPP–TCP BRIDGE

↓

BRIDGE WRAPS IFPP PACKET IN TCP PAYLOAD

↓

SEND VIA SOCKET

↓

REMOTE BRIDGE UNWRAPS → PASSES TO LIAISON

Behavior:

- Maintains socket only within bridge layer.
- Transitional mode for legacy internet routes.

Analogy:

A pigeon carrying a USB stick — TCP/IP carries IFPP intact through legacy space.

A.7. UNIFIED EVENT-DRIVEN MODEL

ALL TRANSPORTS FOLLOW THE SAME MINIMAL EVENT LOOP:

```
DEF ON_CANDIDATE_DETECTED(DEVICE):
    IF VERIFY_DIGEST(DEVICE):
        EMIT("HANDSHAKE_READY", DEVICE)

DEF ON_HANDSHAKE_READY(DEVICE):
    EPH_KEY = POINT_ANGEL.CREATE_EPHEMERAL_KEY(DEVICE)
    LIAISON_ANGEL.TRANSMIT(DEVICE, BUILD_IFPP_PACKET(EPH_KEY))

DEF ON_PACKET_RECEIVED(PACKET):
    IF VERIFY_PACKET(PACKET):
        TEAM_LEADER_ANGEL.DECIDE_NEXT_ACTION(PACKET)
```

A.8. COMPARATIVE TRANSPORT TABLE

Mode	Carrier	Persistent?	Energy	Typical Use	Notes
UDP	*Datagram*	❌	*Medium*	*Internet hops*	*Stateless, self-authenticating*
Bluetooth	*BLE/GATT*	❌	*Low*	*Local mesh / disaster*	*No pairing required*
Wi-Fi	*Mesh/Direct*	❌	*Medium*	*Swarm propagation*	*IP-optional*
TCP/IP Bridge	*Socket*	✅ *(bridge only)*	*High*	*Legacy link*	*Transitional support*

A.9. DEVELOPMENT STATUS & FUTURE REVISION

THIS ANNEX DEFINES THE CURRENT WORKING MODEL OF IFPP'S MESSAGE PACKET AND STATELESS TRANSPORT LOGIC.

The system remains under active development and will evolve with future validation tests on swarm behavior, device-level energy dynamics, and AI-driven optimization.

Planned developments include:

1. *Adaptive swarm load-balancing*
2. *Energy-aware AI hop selection*
3. *Dynamic encryption key rotation*
4. *Open SDK for encapsulation libraries*

Once additional engineers, interns, and test devices become available, each transport mode will undergo unit testing, benchmarking, and security evaluation.

All results will be integrated into subsequent revisions of this annex.

A.10. REFERENCE MESSAGE JSON STRUCTURES

A.10.1. EPHEMERAL HEADER (EXAMPLE)


```json
{
  "message_id": "sha3-512:...",
  "fractal_hop_id": "A1B3",
  "fractal_depth": 3,
  "parent_hop_id": "A1B",
  "hop_count": 5,
  "source_pubkey_fpr": "ed25519:ab12...",
  "prev_device_pubkey_fpr": "ed25519:cd34...",
  "next_candidate_list": ["ed25519:ef56", "ed25519:gh78"],
  "blob_id": "blob-2025-10-13-0001",
  "manifest_id": "man-2025-10-13-01",
  "bootstrap_nonce_id": "nonce-0001",
  "been_here_digest": "sha3-256:4a3f...",
  "ai_digest": "sha3-256:9f1b...",
  "hop_hash_seed": "sha3-256:7z...",
  "sacred_persistence": true,
  "flags": { "ack_requested": true, "push_mode": false },
  "signature": "ed25519:..."
}
```

A.10.2. PAYLOAD SIDECAR (EXAMPLE)


```json
{
  "blob_id": "blob-2025-10-13-0001",
  "message_id": "sha3-512:...",
  "content_type": "text/plain",
  "pre_encryption_digest": "sha3-256:...",
  "encryption": {
    "algorithm": "XChaCha20-Poly1305",
    "key_id": "sesh-2025-10-13-01",
    "nonce": "..."
  },
  "display_header": { "from_label": "Alice", "to_label": "Bob" },
  "fractal_seed": "A",
  "integrity_core_hash": "sha3-512:..."
}
```


A.10.3. AI HOP DELTA (EXAMPLE)


```json
{
  "message_id": "sha3-512:...",
  "current_fhi": "A1B3",
  "prev_fhi": "A1B2",
  "device_pubkey_fpr": "ed25519:cd34...",
  "hop_perf_delta": { "latency_ms": 22, "rssi":-56 },
  "integrity_flag": true,
  "learning_delta_digest": "sha3-256:..."
}
```

A.11. CLOSING SUMMARY

Annex A captures the current formal definition of IFPP's stateless message and transport framework.

As development continues, these specifications will serve as the canonical baseline for interoperability testing, hardware adaptation, and swarm-level AI performance research.

Future revisions may alter field naming, hashing algorithms, and routing logic based on swarm telemetry.

The IFPP team encourages external collaboration — from engineers, testers, and academic partners — to refine, validate, and expand this model into its production-ready form.

A.12 — STORAGE & BANDWIDTH PROJECTION MATRIX

Component	Avg Size (Bytes)	Growth per Hop	Retention	Notes
Ephemeral Header	*700–1,200*	*+10–50*	*Transient*	*Signed JSON, volatile*
AI Metadata (delta)	*512–1,024*	*+100–200*	*Per-hop*	*Small digest + learning vector*
AI Metadata (full local)	*2–5 KB*	*+100–200*	*Until message completion*	*Retained for traceback & swarm learning*
Eternal Message Core (payload)	*10 KB–10 MB*	*None*	*Persistent*	*Encrypted payload*
Aggregate Hop Packet	*1.5–2.5 KB*	*+~2%*	*Transient*	*On-wire size*
Device Storage (avg 500 msgs)	*10–200 MB*	*Linear*	*Configurable*	*Depends on payload policy*

Efficiency Comparison *(per 10 KB payload):*

- ***TCP/IP (SMTP EQUIVALENT)****: ~14 KB transfer cost (headers, ACKs, handshake).*
- ***IFPP: ~11.2 KB*** *total including metadata, no session overhead.*
  → ~20–25% lower total cost under typical swarm load.

Scaling Projection *(per 1MB, 1000 devices):*

- *Estimated 1.1 GB total swarm footprint (with redundancy ×3).*
- *Scales fractally; propagation throttles after 5th generation (Sacred Persistence threshold).*

A.13 – *INTEGRITY AND TRUST MATRIX*

THE IFPP PROTOCOL EMPLOYS MULTI-LAYERED AUTHENTICATION AND CRYPTOGRAPHIC PROOFS DISTRIBUTED ACROSS THE MESSAGE PACKET, DEVICES, AND SWARM NODES. EACH PART CONTRIBUTES UNIQUELY TO MESSAGE TRUST, WITHOUT REQUIRING CENTRAL AUTHORITY.

Layer	*Function*	*Verification Source*	*Key Used*
Ephemeral Header	*Hop authenticity & lineage*	*Device signature per hop*	*Ed25519 (hardware key)*
AI Metadata	*Swarm learning & traceability*	*Gabriel's integrity digest*	*SHA3-256 + Bloom hash*
Eternal Message Core	*Payload immutability*	*Source/destination swarm keys*	*XChaCha20-Poly1305 + Ed25519*
Device Identity	*Node-level attestation*	*Platform keystore (TPM/TEE)*	*Non-exportable key slot*
Swarm Integrity	*Collective state verification*	*Gabriel's network root hash*	*Rotating root keychain*
Revocation/Updates	*Trust pruning & key rotation*	*Signed manifest via Gabriel*	*Multi-signed ledger entry*

Key Principle:

INTEGRITY *emerges from* ***DISTRIBUTED VERIFICATION*** *— each hop confirms authenticity without needing to trust previous ones, yet all are traceable through Gabriel's transparent, immutable digests.*

A.14 DEVELOPER IMPLEMENTATION SUMMARY

THIS SECTION OUTLINES THE RECOMMENDED DEVELOPER ARCHITECTURE AND INTEGRATION APPROACH FOR IMPLEMENTING THE IFPP MESSAGE FORMAT WITHIN APPLICATIONS, EMBEDDED DEVICES, OR RELAY SYSTEMS.

A.14.1 Language and Platform Bindings

Layer	***Target Platform***	***Recommended Language***	***Notes***
Core IFPP Library	*Linux / Android*	*Python, Rust*	*Reference implementation*
Relay Engine	*Android / Embedded*	*Kotlin, C++*	*Integrates device transports*
AI Metadata Handling	*Server / Edge*	*Python (NumPy / PyTorch)*	*Federated learning & digest*
Verification / Audit	*Any*	*Rust or C*	*Deterministic verification tools*
Layer	*Target Platform*	*Recommended Language*	*Notes*
Core IFPP Library	*Linux / Android*	*Python, Rust*	*Reference implementation*

A.14.2. Canonical Serializer

- *Format: Deterministic **JSON** or CBOR*
- *Sorting: **ALPHABETICAL KEY ORDER***
- *Signature Coverage: **ENTIRE HEADER** (excluding signature field)*
- *Hashing: SHA3-512 for **message_id**, SHA3-256 for **ai_digest***

A.14.3. *Core Function Prototypes (IFPP SDK Reference)*

```python
# Core creation
def create_message(payload_bytes, to_pubkey, options) -> message_id: ...
def build_ephemeral_header(message_id, blob_id, fractal_seed, manifest_id) -> dict: .

# Signing and verification
def sign_header(header_dict) -> str: ...
def verify_header(header_dict, pubkey_fpr) -> bool: ...

# Propagation
def perform_handshake(target_device) -> SessionKey: ...
def send_packet(header_dict, transport_mode) -> bool: ...
def forward_packet(header_dict, new_targets) -> None: ...

# AI Metadata
def append_ai_metadata(message_id, hop_delta) -> None: ...
def generate_ai_digest(message_id) -> str: ...

# Audit and verification
def export_mission_bundle(message_id) -> tarball: ...
```

A.14.4. *Implementation Notes*

- ***PAYLOAD FRAGMENTATION***: *handled automatically by IFPP library (BLE MTU-aware).*
- ***HOP STATE STORAGE***: *each device retains a minimal "mission envelope" until acknowledgment.*
- ***KEY MANAGEMENT***: *device keys remain sealed within secure enclave; Gabriel verifies hashes externally.*
- ***API EXPOSURE***: *developers interact only through high-level functions (Team Leader → Liaison).*

A.15. VISUALIZATION SUMMARY

FIGURE A.1 — IFPP THREE-PART MESSAGE FLOW

```
+-----------------------+        +-----------------------+        +-----------------------+
| Device A              |        | Device B              |        | Device C              |
|  (Team Leader)        |        |  (Intermediate)       |        |  (Recipient)          |
+-----------------------+        +-----------------------+        +-----------------------+
| Ephemeral Header (Seed) | ---->  | Verifies + re-signs   | ---->  | Confirms + deletes temp |
| AI Metadata (Delta)   |        | Appends local digest  |        | Updates full metadata |
| Eternal Core (Payload) |        | Forwards Core reference |        | Decrypts Core (SRK)   |
+-----------------------+        +-----------------------+        +-----------------------+

             ↑
     Gabriel monitors digests and hashes transparently (non-invasive)
```

A.16. CLOSING STATEMENT

WITH THESE EXTENSIONS, THE IFPP PACKET STRUCTURE IS NOW A COMPLETE, STANDALONE TRANSPORT FRAMEWORK, EQUIVALENT IN CLARITY AND FUNCTION TO TCP/IP PACKET DEFINITIONS, BUT MODERNIZED FOR DISTRIBUTED, STATELESS AI-DRIVEN PROPAGATION.

It provides:

- *A uniform message schema (Ephemeral, AI, Core).*
- *Clear verification logic per layer.*
- *Quantitative metrics for bandwidth, energy, and persistence.*
- *Developer references and APIs for practical implementation.*

This annex, together with the IFPP whitepaper, serves as the canonical technical reference for anyone seeking to implement or extend MAMAWMAIL's intelligent fractal propagation system.