Transcribed Technical Specification: Distributed System Error Model

Error Level Architecture for Distributed Systems (-1 to -12)

Core Concept: Implement distributed system error handling with AVL-Huffman based node rotation for fault tolerance, where each error level triggers isomorphic rotations of AVL nodes with phenomenological instance continuity.

Network Architecture

```
<reference_to_peer_mode>
peer_node = {
    network_node: IP_address,
    polyglot_port_mapping: service_ports,
    peer_services: [...]
}
```

Bidirectional Recovery Protocol: Shared states for services with fault tolerance cost metrics. Using wildcard patterns `*` for any program language extension (`.py`, `.pyc`, etc., including Cython effort bounds).

Error Level Classification

Normal Operation:

- `0`: No errors, exceptions, or panics

Warning Distribution Scheme (-1 to -3):

- Low to high warning levels
- Distribution state unit and binary handling for two-node systems

Danger Levels (-4 to -6):

- Low to high danger states
- Distribution based on schema

Critical System Danger (-7 to -9):

- Low to high critical danger for system

System Kill States (-10 to -12):

- Kills program node in peer-to-peer mode

- Based on Byzantine fault tolerance model
- Kills system state for hijack extraction vectors

Positive Error States (1 to 12)

For development to production CI/CD integration when human is "actively in the loop" - building, testing, documenting, developing instances of `gosi.exe`.

Implementation Notes

- All errors/exceptions/panics are handled smoothly to stop passive system degradation
- Peer-to-peer nodes maintain network connectivity through IP addresses and polyglot port mappings
- Service-based I/O fault tolerance with cost metrics for two main components
- Wildcard support for multiple programming language extensions

This aligns with the OBINexus fault-tolerant distributed systems framework using category theory, where fault states guide system responses through graduated witnessing membranes.