0.1 Ants, Bees, Snakes, and Worms

Came from 2025-papers/Topology-Addressing/Topology-Addressing.tex

A sea in IPUs, within Rack-Scale and Chiplet Interconnects require additional levels of abstraction for configuration and reconfiguration than is provided in existing Standards. This paper Explores biologically inspired topology-learning and routing methods in networks of 8-bidirectional port nodes, with a special focus on wormhole routing and its variants. We then take a look at current state-of the art routing technologies, such as deflection forwarding combined with local compass point addressing and see how they match.

Modern system-on-chip designs increasingly rely on chiplet-based architectures, where multiple specialized chiplets—each dedicated to a particular function (e.g., CPU, GPU, accelerator)—are connected to form a unified and scalable platform. Ensuring that these chiplets communicate efficiently poses a variety of challenges, especially as the number of chiplets grows and each chiplet (or on-chip router) may have multiple ports (valencies of 4, 6, or 8).

Existing solutions often rely on standard routing protocols or static configurations. However, as topologies become denser, more heterogeneous, and subject to partial reconfiguration, novel methods are needed to learn, update, and maintain the interconnect topology. Drawing inspiration from biological systems—where ants, bees, snakes, spiders, and now worms—each exhibit unique strategies for exploration, organization, and adaptation, promises fresh ideas for topology discovery, addressing, and low-level routing mechanisms.

0.1.1 Challenges in Chiplet Interconnects

1. Scalability: As the number of chiplets increases, the complexity of maintaining accurate global network knowledge grows exponentially. Conventional centralized or static approaches can struggle to stay current and efficient.
2. Partial Knowledge: On-chip routers typically have limited local visibility. They need to cooperate with neighbors to build a broader perspective of the network.
3. Adaptive Reconfiguration: Certain chiplets may power down, enter low-power states, or be repurposed. The routing and addressing approach must cope with these dynamic behaviors.
4. Latency Sensitivity: On-chip communications can be highly latency-sensitive.Any topology exploration or routing technique must have minimal overhead in time and power.
5. Physical Constraints: Depending on the routing mechanism (e.g., store-and-forward, cut-through, or wormhole routing), the buffer sizing, FIFOs, and local SRAM usage become critical design con-

siderations.

Biologically inspired approaches have shown promise in large-scale or dynamic networks because they emphasize local decisions and emergent global behaviors, while specialized routing methods at the flit or byte level can significantly impact network performance and resource usage.

0.1.2 1. Ants: Stigmergic Discovery & Adaptive Routing

In nature, ants communicate via pheromones—chemical trails used to discover and reinforce paths to resources. In a chiplet context, ant-inspired algorithms involve agents (packets) that:

1. Randomly Explore: "Scout ants" are periodically sent into the network to discover unknown or less-traveled routes.
2. Deposit "Pheromones": As these scouts move, they leave "digital pheromones" in local tables, indicating path quality or latency.
3. Positive Feedback: Over time, successful routes are reinforced, prompting data packets to favor paths with strong pheromone levels.
4. Decay Mechanism: Pheromones degrade, allowing the system to adapt if congestion or failures cause route performance to change.

0.1.3 Benefits:

- Adaptive to Congestion: Routes are continuously refined by real-time traffic patterns.
- Localized Decisions: Each node handles small amounts of metadata without needing a global map.
- Minimal Hardware Overhead: Requires storing pheromone metrics (e.g., counters, latency measures) in local tables.

0.1.4 2. Bees: Collaborative "Hive" and "Scout" Behavior

Honeybees manage tasks by scouting for resources and returning to the hive to share discoveries. A bee-inspired algorithm can leverage:

1. Hive Nodes: Certain nodes (management chiplets) aggregate topology or performance data.
2. Scout Bees: Packets leaving the hive to explore unknown or under-explored areas, returning with updated route metrics.
3. Recruitment: High-value or high-performance routes are "advertised," encouraging worker packets to follow them.
4. Dance Protocol: Returning scouts might broadcast partial routes or performance gains, influencing how future packets select paths.

0.1.5 Benefits:

- Semi-Centralized Knowledge: Hive nodes maintain or compile partial global knowledge for better load balancing.
- Dynamic Adaptation: Over time, good routes become well-known, while less efficient links see fewer packets.
- Diagnostic Aid: Central nodes can help with debugging or performance tuning.

0.1.6 3. Snakes: Sequential Path Traversals for Comprehensive Mapping

Snakes systematically traverse an environment. In a chiplet network, a snake-inspired approach might:

1. Slithering Packets: A special packet is launched that attempts to traverse every reachable node in a systematic pattern.
2. Topology Collection: As it hops, it collects neighbor lists, port connections, and node identifiers.
3. Loopback: Upon completing a traversal (or hitting boundaries), the packet returns to its origin with a consolidated network map.
4. Distributed Snapshots: Multiple vantage points or repeated "snake sweeps" yield an updated global view of connectivity.

Benefits:

- Guaranteed Coverage: Ensures every node and link is discovered periodically.
- Simplicity: Conceptually straightforward, though it can be heavier in overhead.
- Occasional Diagnostics: Particularly useful for network-wide verification or fault checks.

0.2 4. Spiders: Building and Maintaining "Webs" of Connectivity

Spiders create webs that dynamically adapt to external stresses or breaks. A "web-based" approach for chiplet networks focuses on building a resilient mesh:

1. Web Construction: Each router sends out "threads"—short discovery packets—on all ports. Neighbors respond, forming local connectivity data structures.
2. Local Weave: Threads intersect and overlap, letting routers learn about multi-hop neighbors.
3. Damage Repair: If a link fails, local threads are resent to repair or reroute around the break.
4. Tension Metrics: Each link in the web holds a "tension" (latency,

throughput) that can be monitored and used to shift traffic if congestion or errors rise.

Benefits:

- Resilience Through Redundancy: Overlapping “threads” ensure multiple known paths.
- Incremental Updates: Each node refines its local web structure.
- Ease of Local Addressing: Short IDs can be assigned to neighbors, aggregated as the web extends outward.