A Dynamic Hierarchical System for Large Scale Distributed Applications

Junichi Uekawa	Grad. School of Doshisha University
Mitsunori Miki	Doshisha University
Tomoyuki Hiroyasu	Doshisha University
Yusuke Tanimura	Grad. School of Doshisha University

Agenda

- DNAS architecture
- Messaging system
- API
- Application: GA
- Results

Background

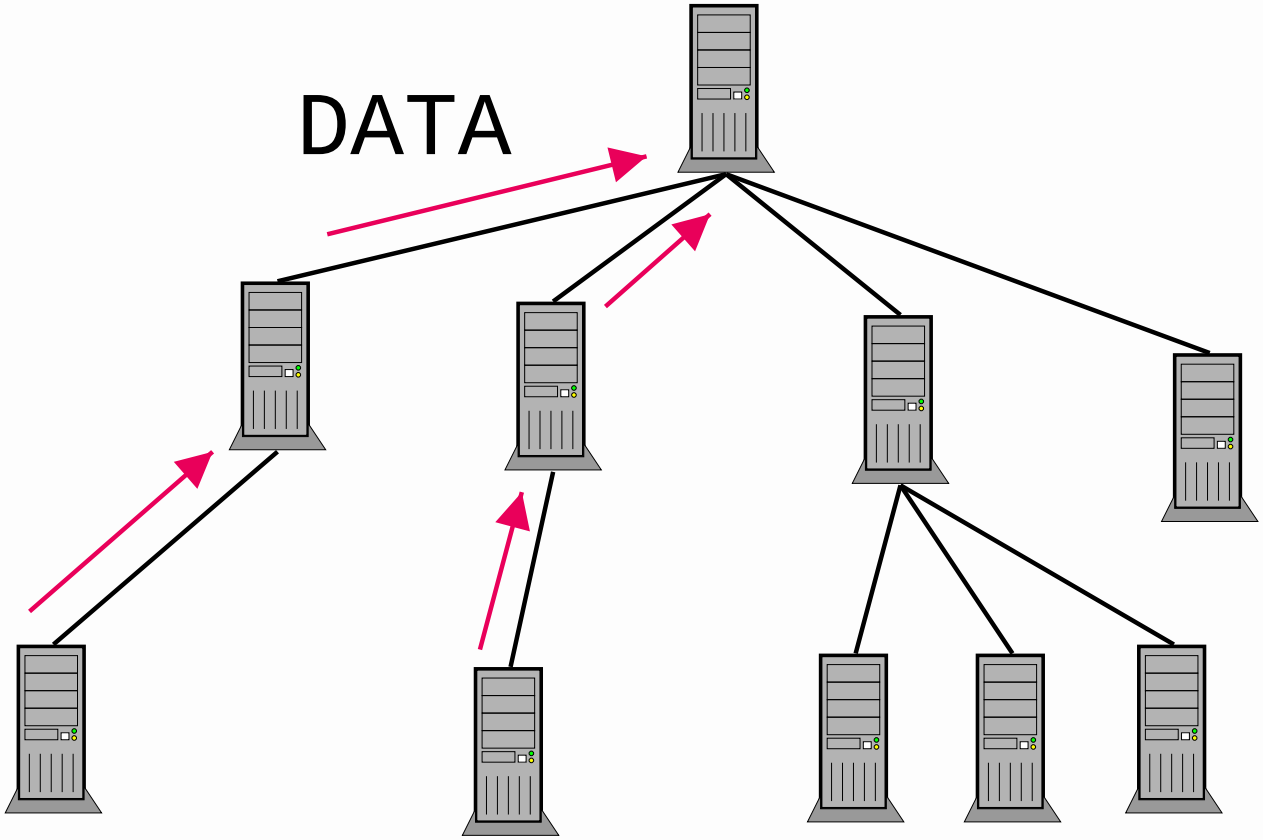
- Massively Parallel Computing
- Grid Oriented Applications
  - Divisible independent jobs
  - Low communication needs
  - Low cost in node change

DNAS

- Distributed Network Application System
- Message passing interface
  - Infrequent communication
  - Transparent node failure mechanism
- Hierarchical structure
  - Efficient local communication
  - Lower load per host

Topology of DNAS

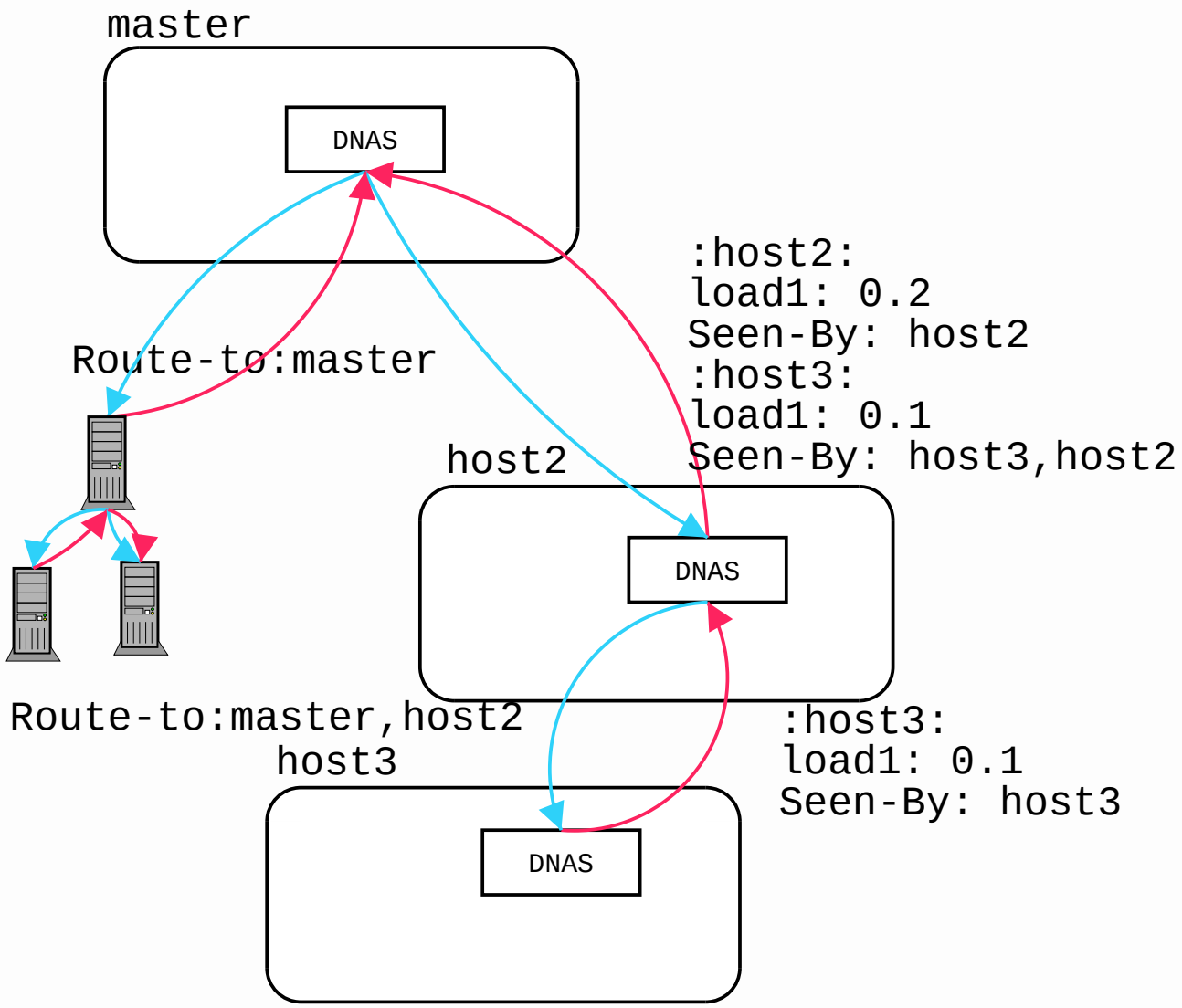
Once every 3 seconds,
automatic hierarchical communication



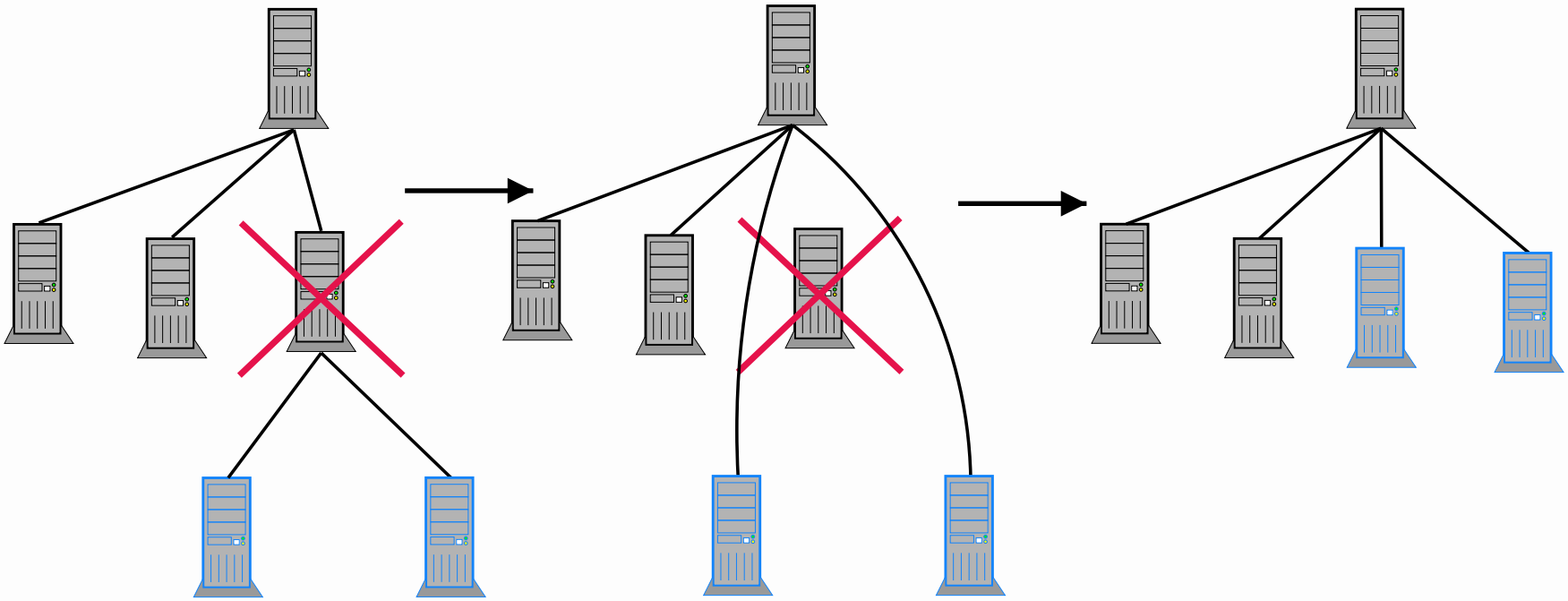
Structure Information

- Seen-By: which hosts a message packet was sent through
- Route-To: The route to the uplink.
- Data-Seen: tags the packet as seen, to avoid loops

Session between two hosts

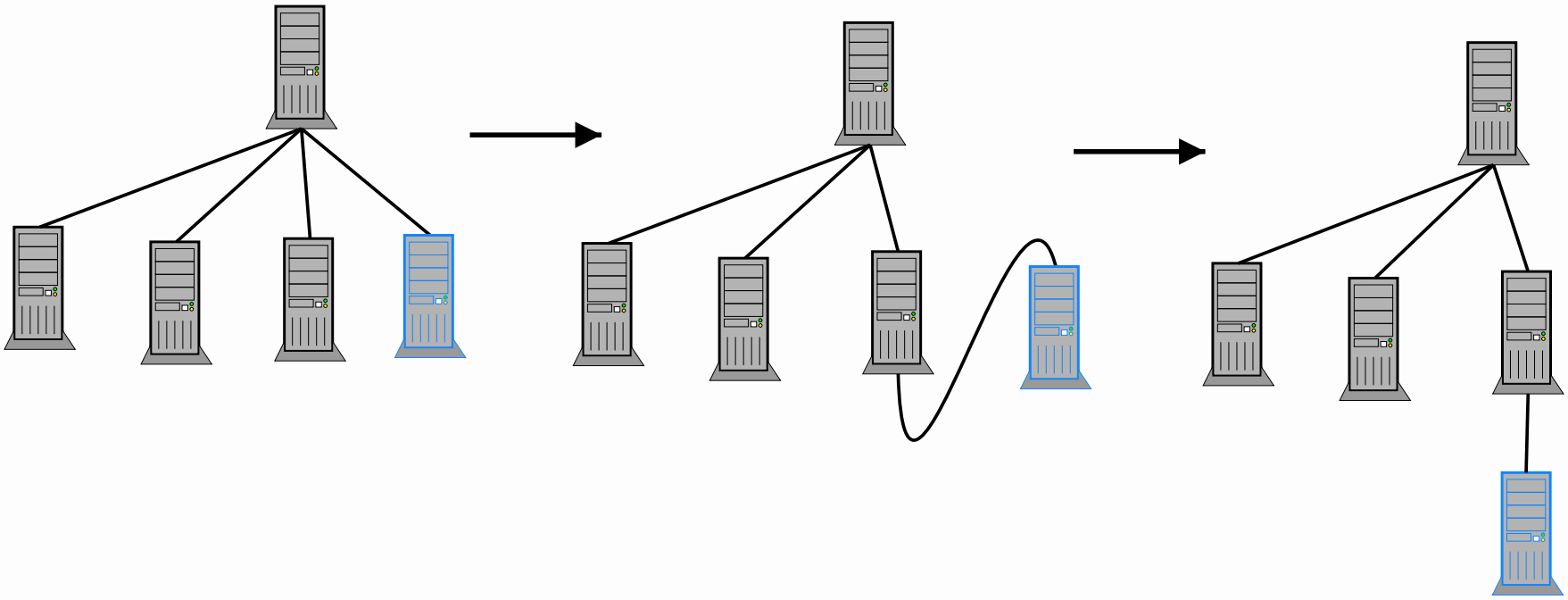


Host re-routing



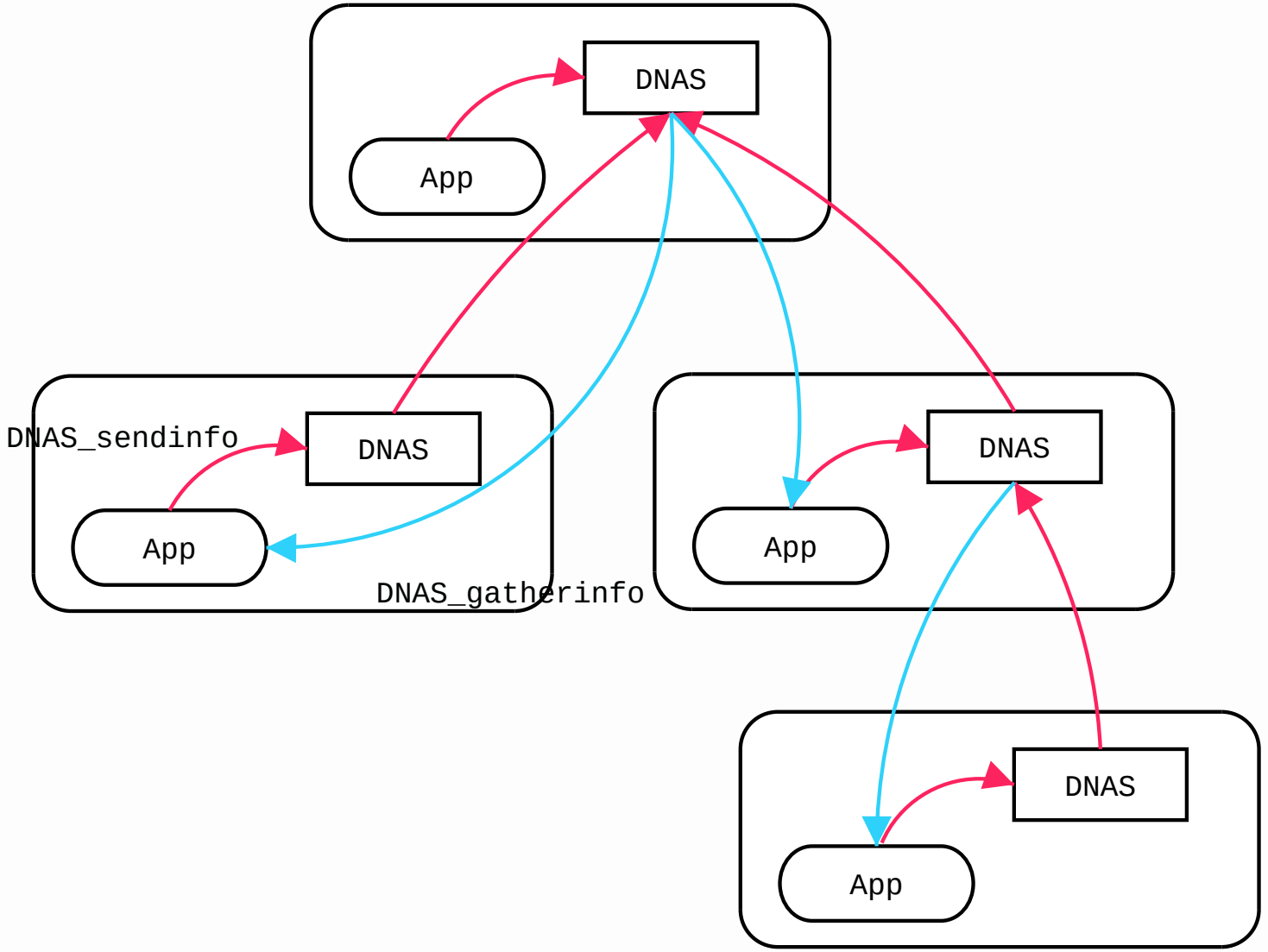
Re-route around removed host

Link-limiting system

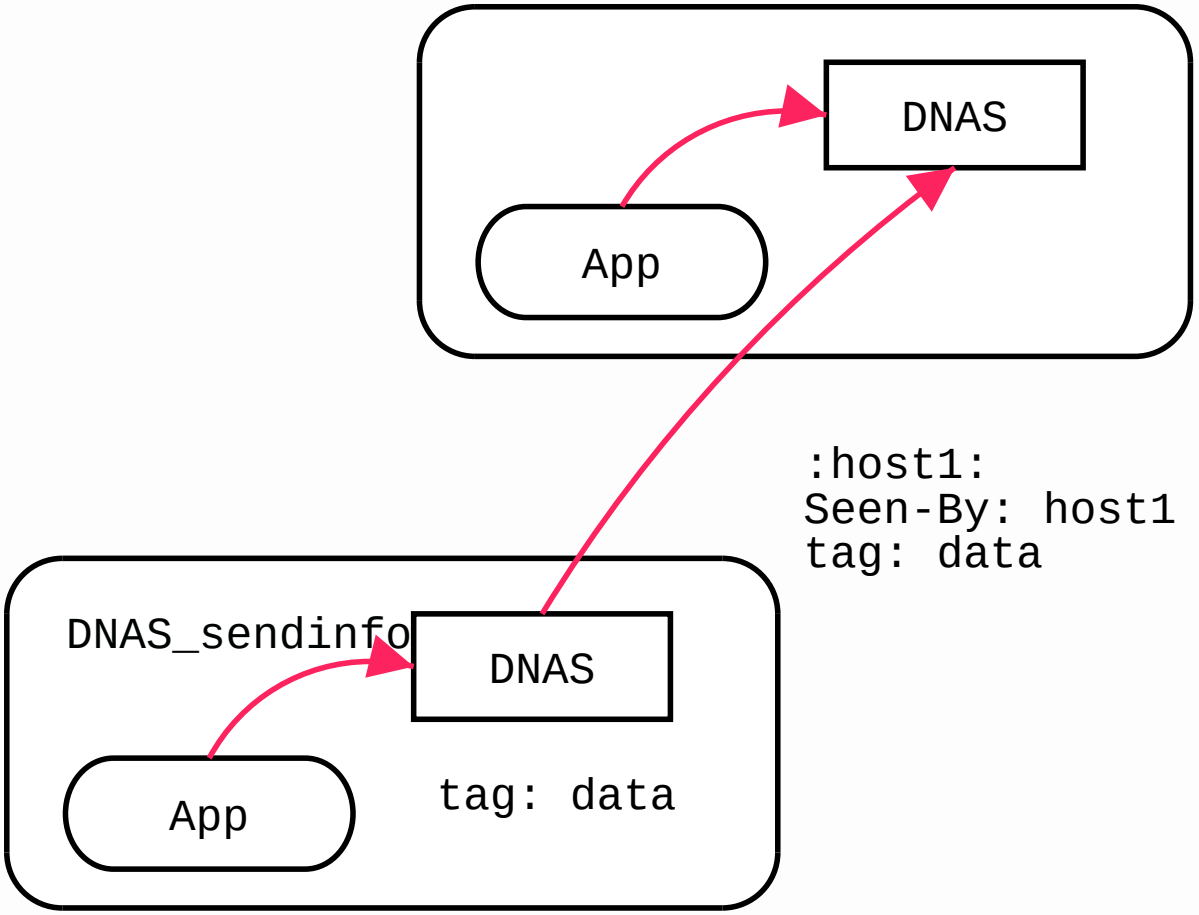


Limit to 3 hosts, specify where to reconnect.

DNAS API


DNAS
App
DNAS_sendinfo
DNAS
App
DNAS_gatherinfo
DNAS
App
DNAS
App

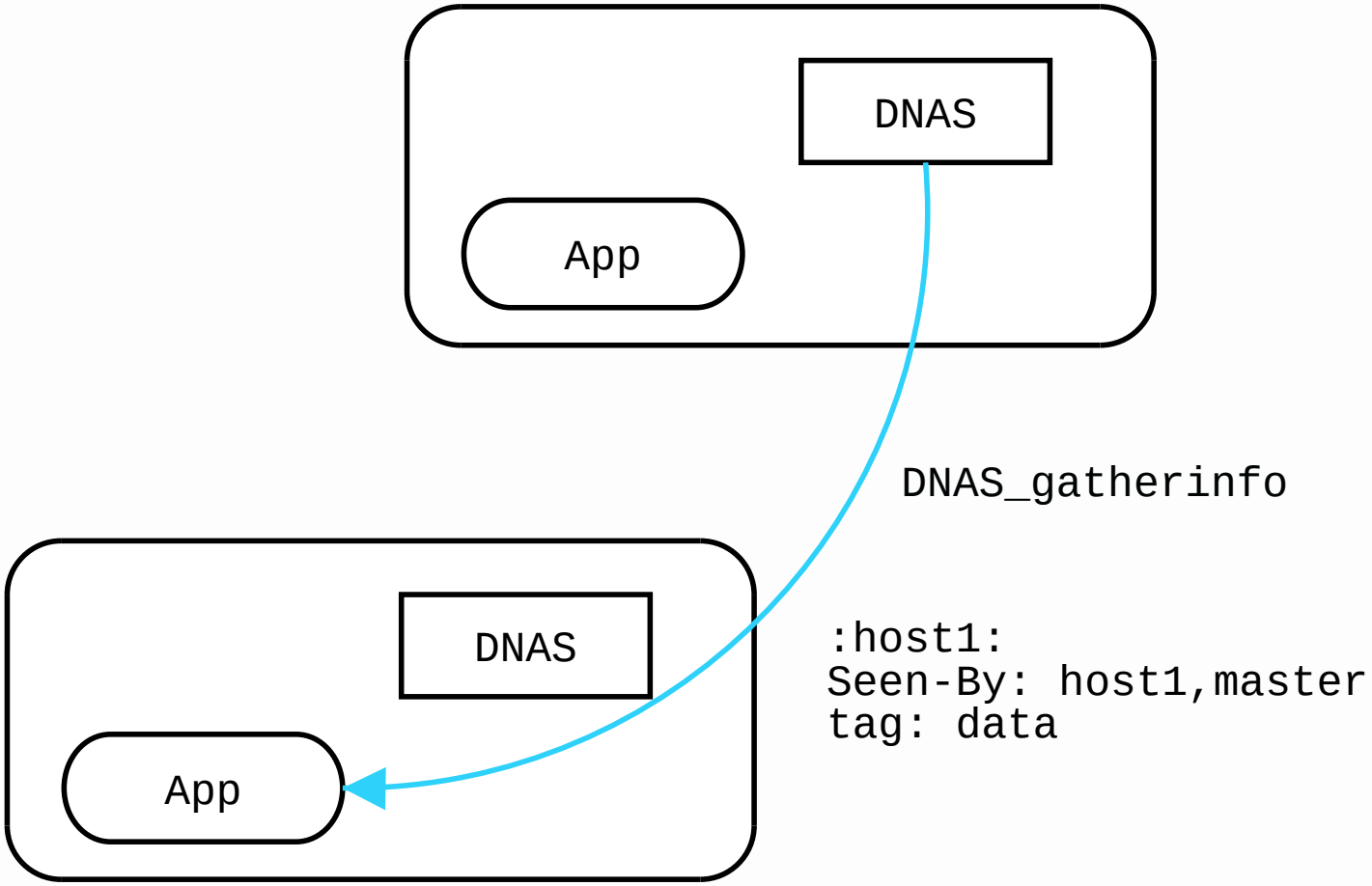
DNAS_sendinfo



Use of tags

- Send to uplink DNAS with TAG “ga-gene”
  - ⇒ Stored as:
    Host name: “ga-gene” data
- Other nodes obtain with “ga-gene” as key

DNAS_gatherinfo



Characteristic of the system

- Relinkage
  - Does not break if nodes are removed
  - Handles new nodes
- Messaging
  - Individual nodes are independent
  - Not knowing what other nodes exist in system

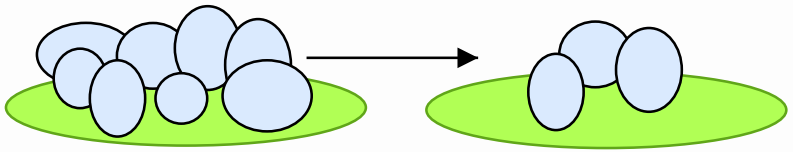
Application

- Implemented solver for Optimization Problems.
  Genetic Algorithm
- Experiments
  - Experiment Environment
    256PE Pentium III 800MHz
    Interconnected with 100BASE-TX

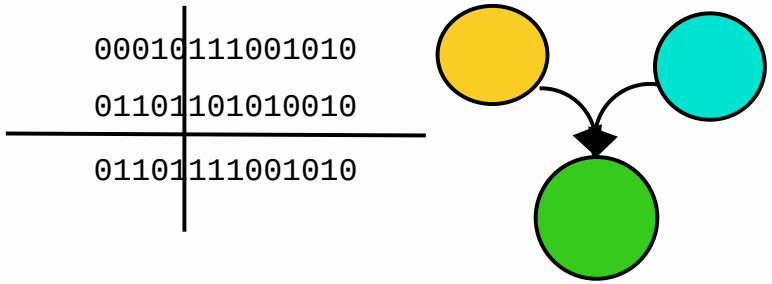
Genetic Algorithm (GA)

- Searching method inspired from Evolution process.
- Design variables of optimization problems are used as genes, and genetic operations; selection, cross over, mutation are done over individuals. Individuals that are more fit remain after a generation.
- After generations, a solution with high fitness comes.

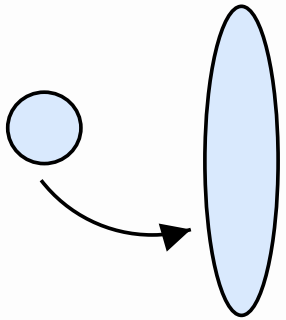
Genetic Algorithm (GA)



Selection: Only the fit survive

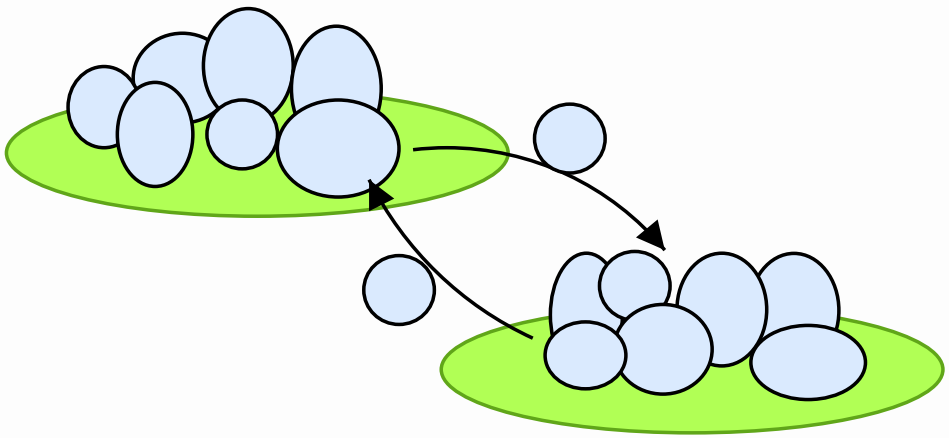


Crossover: combine two partial solutions to generate new solution



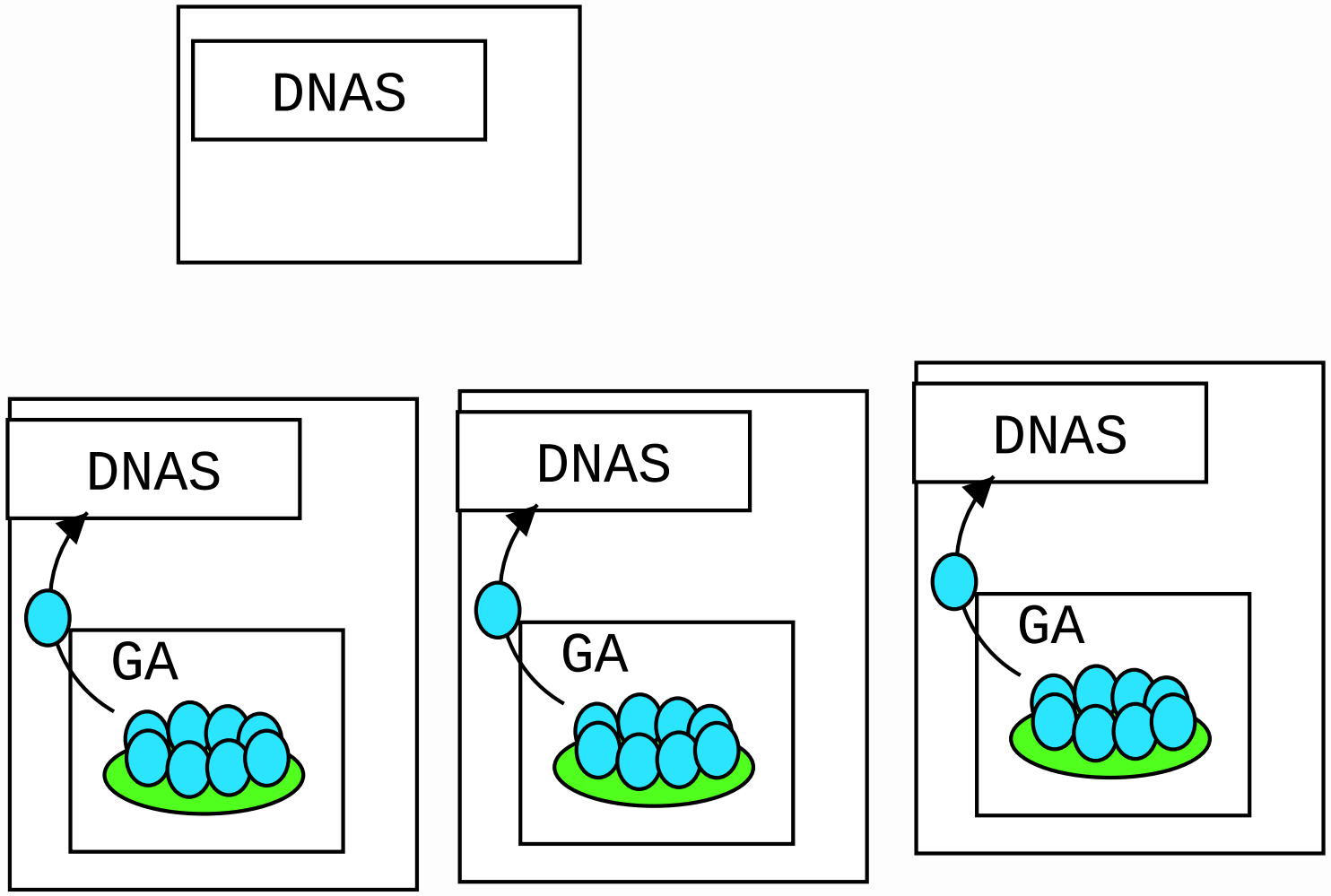
Mutation: Randomly modify individuals

Distributed GA (DGA)

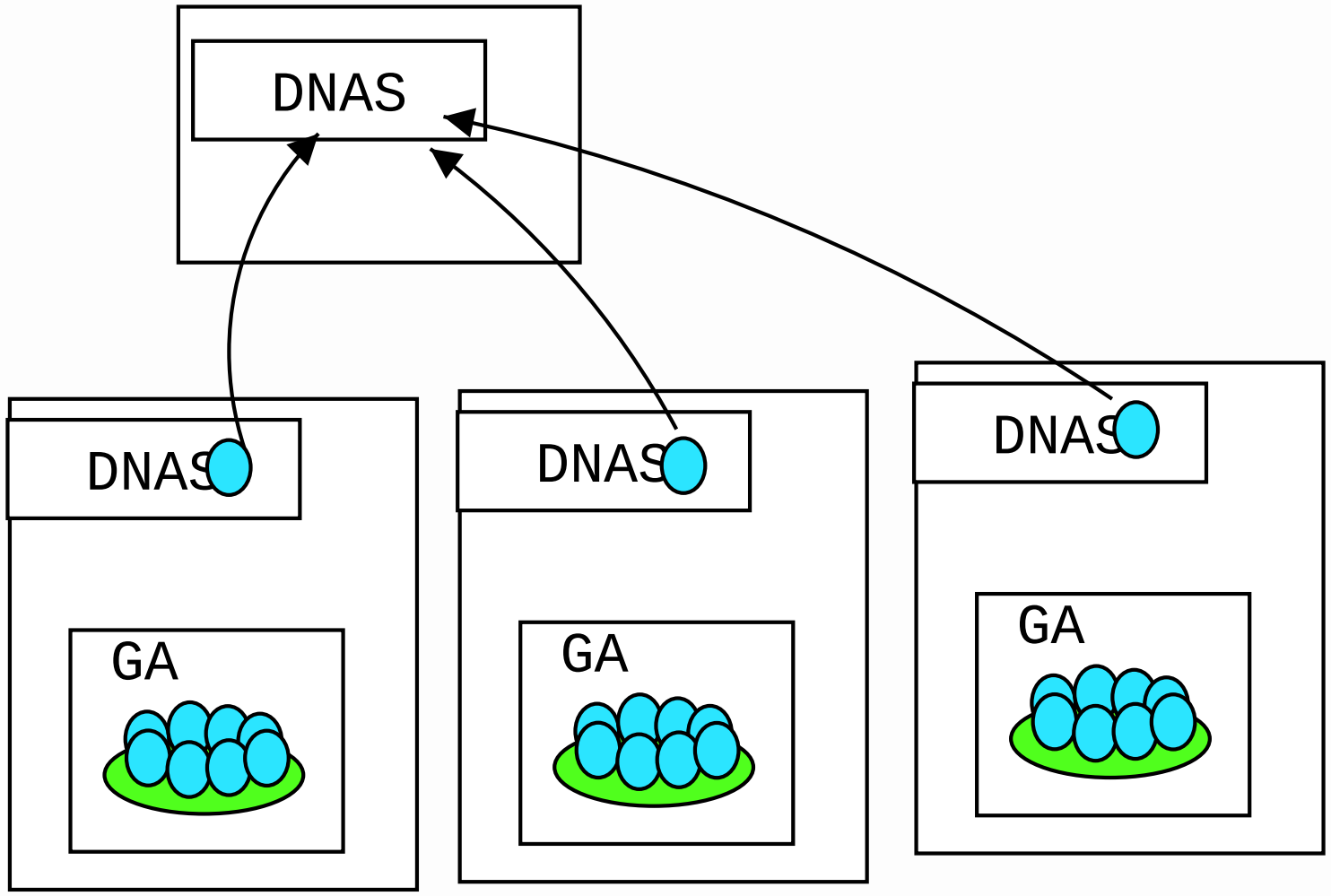


Migration: Exchange individuals between independent islands

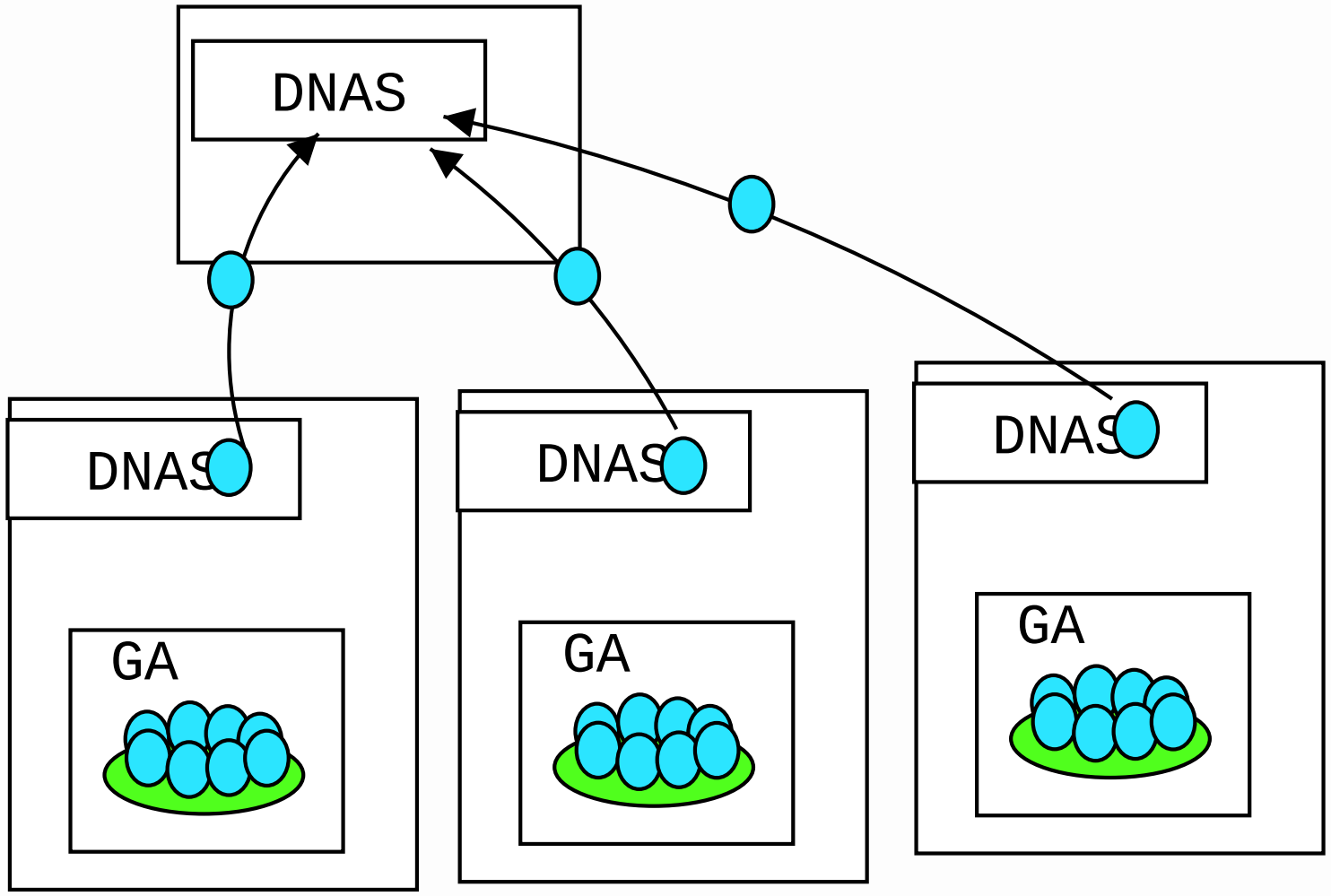
DNAS DGA



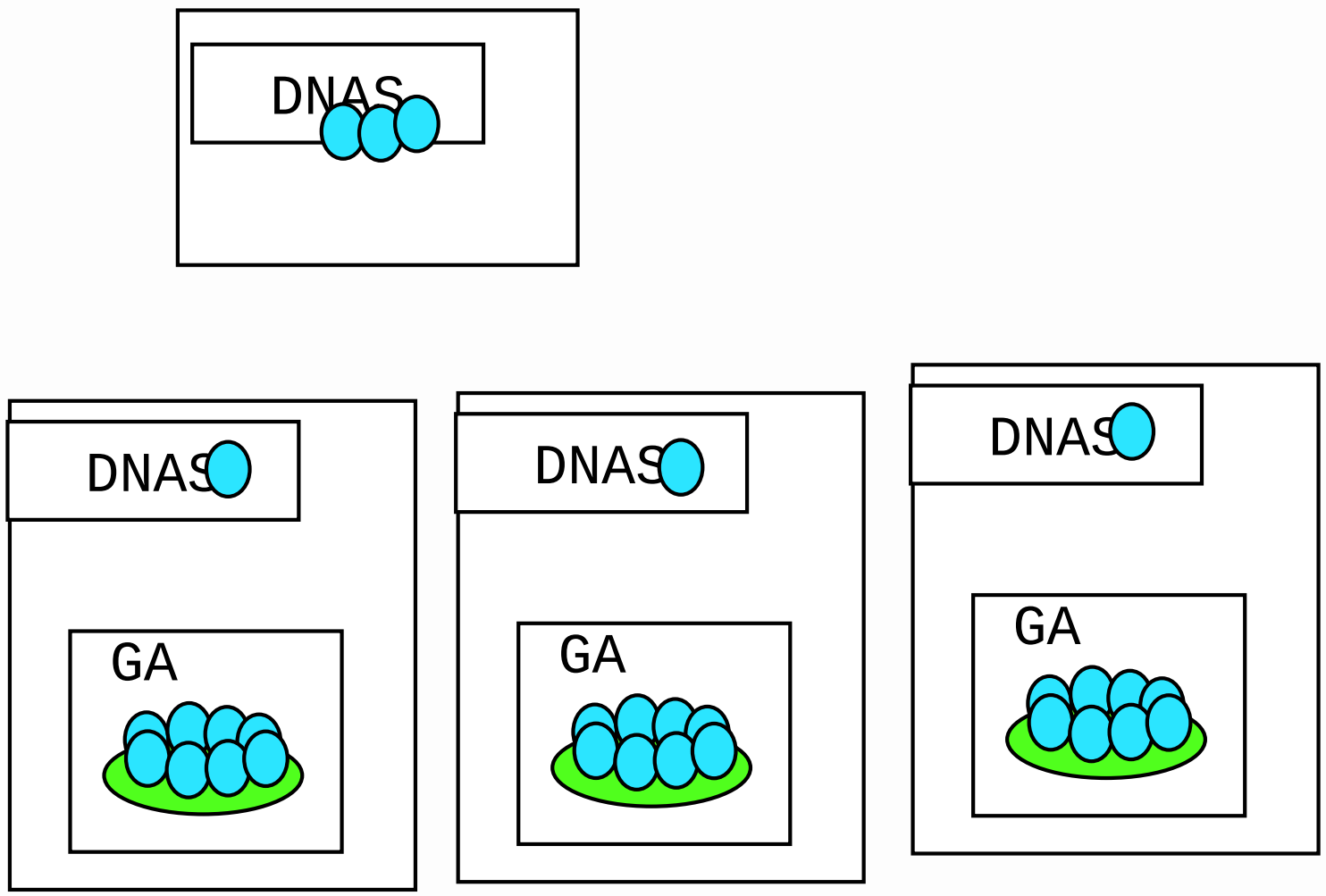
DNAS DGA



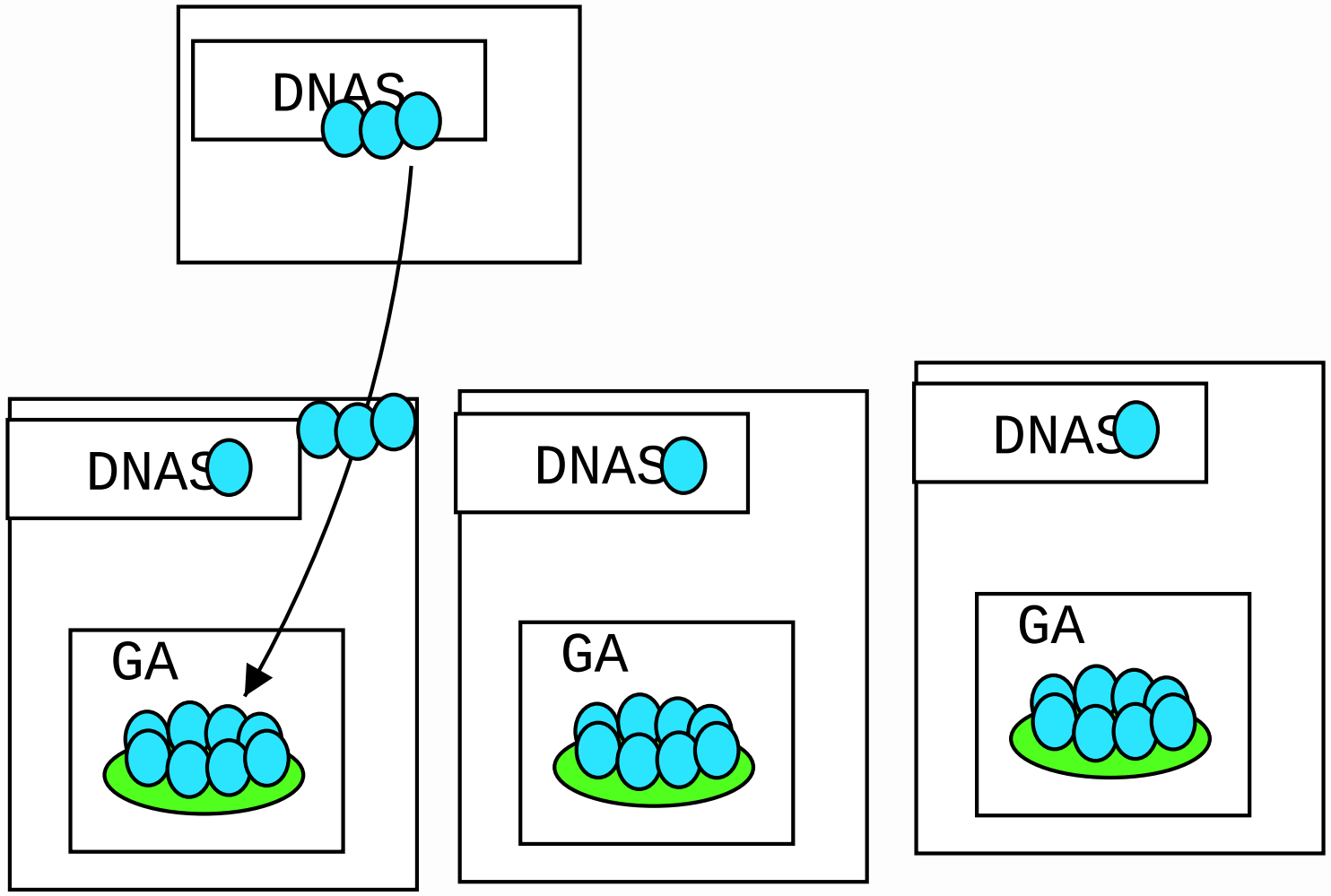
DNAS DGA



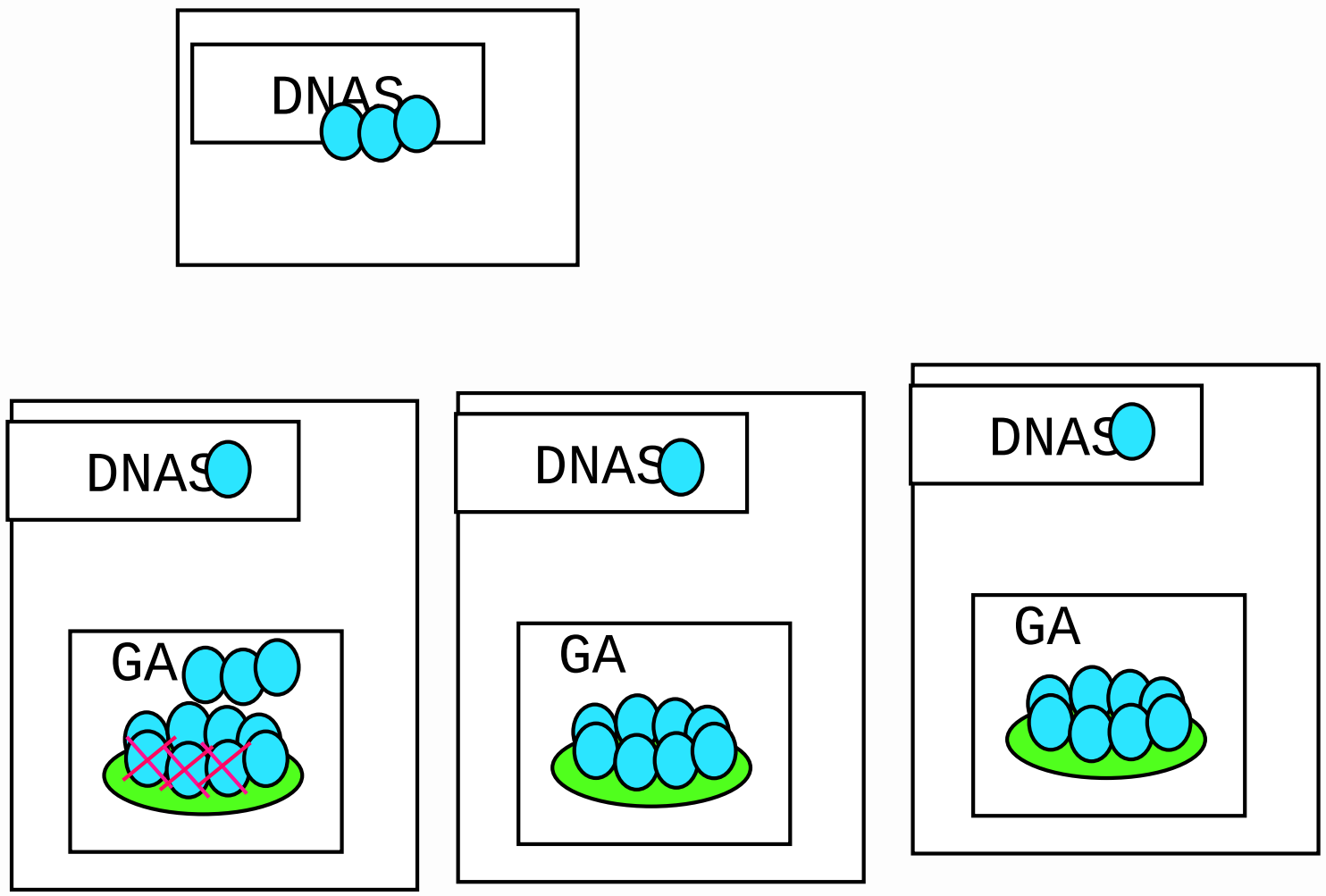
DNAS DGA



DNAS DGA



DNAS DGA



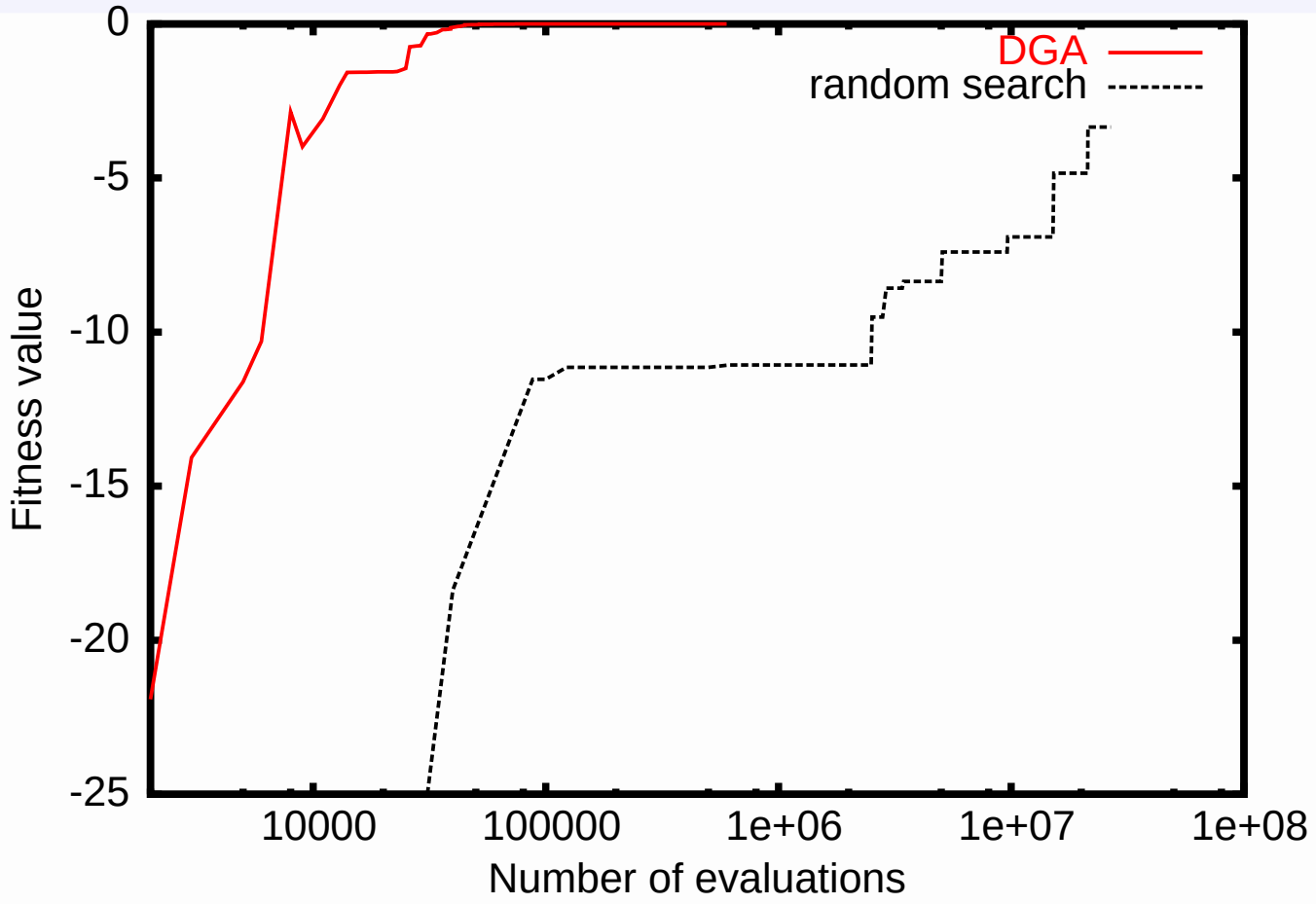
DNAS DGA

- Problem space is 160 bits
- Solution has fitness of 0
  Rastrigin function

$$f = -\left(10n + \sum_{1}^{n} x_i^2 - 10cos(2\pi x_i)\right)$$

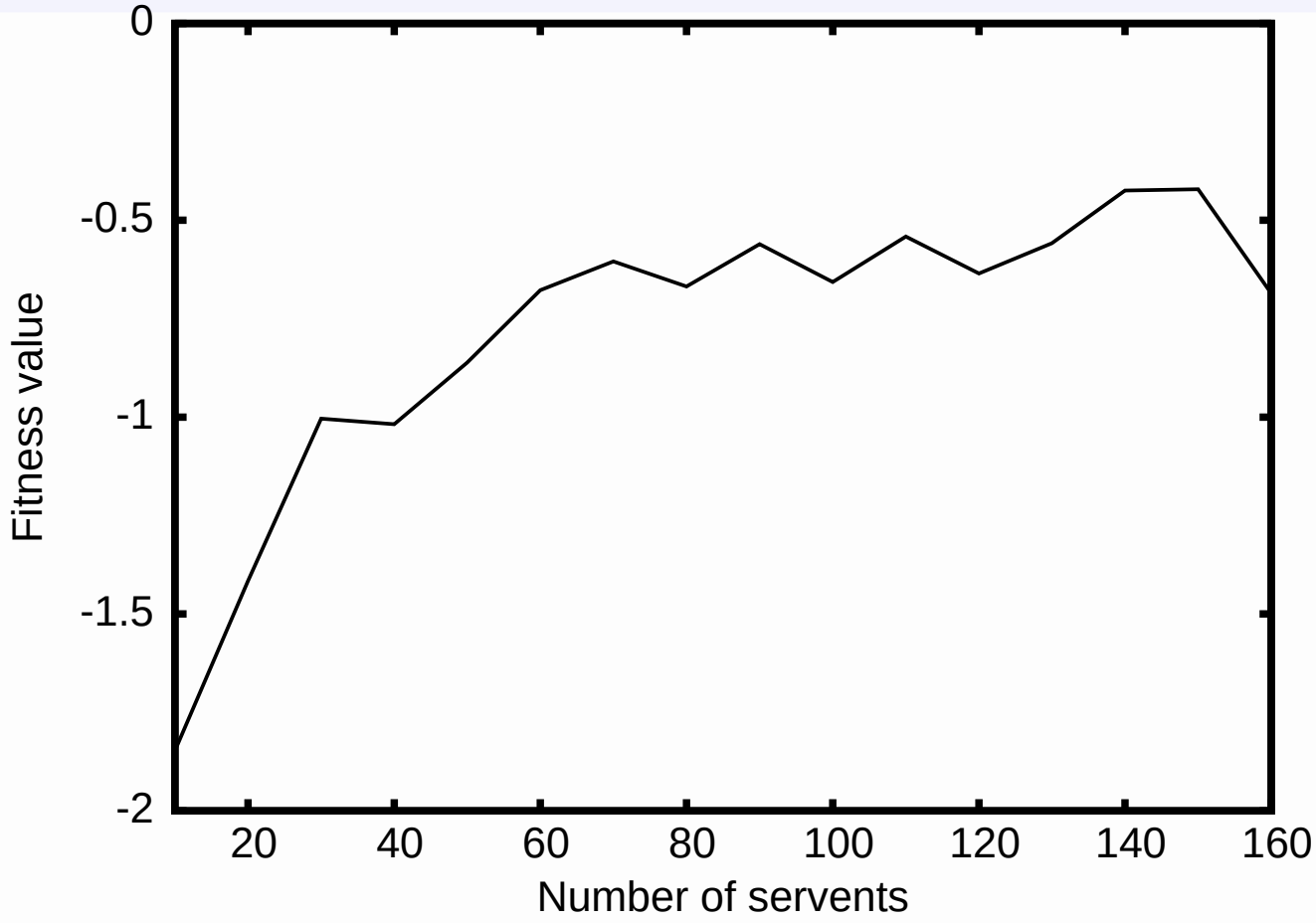
$$x \in [-5.12, 5.12]$$

DNAS GA over random



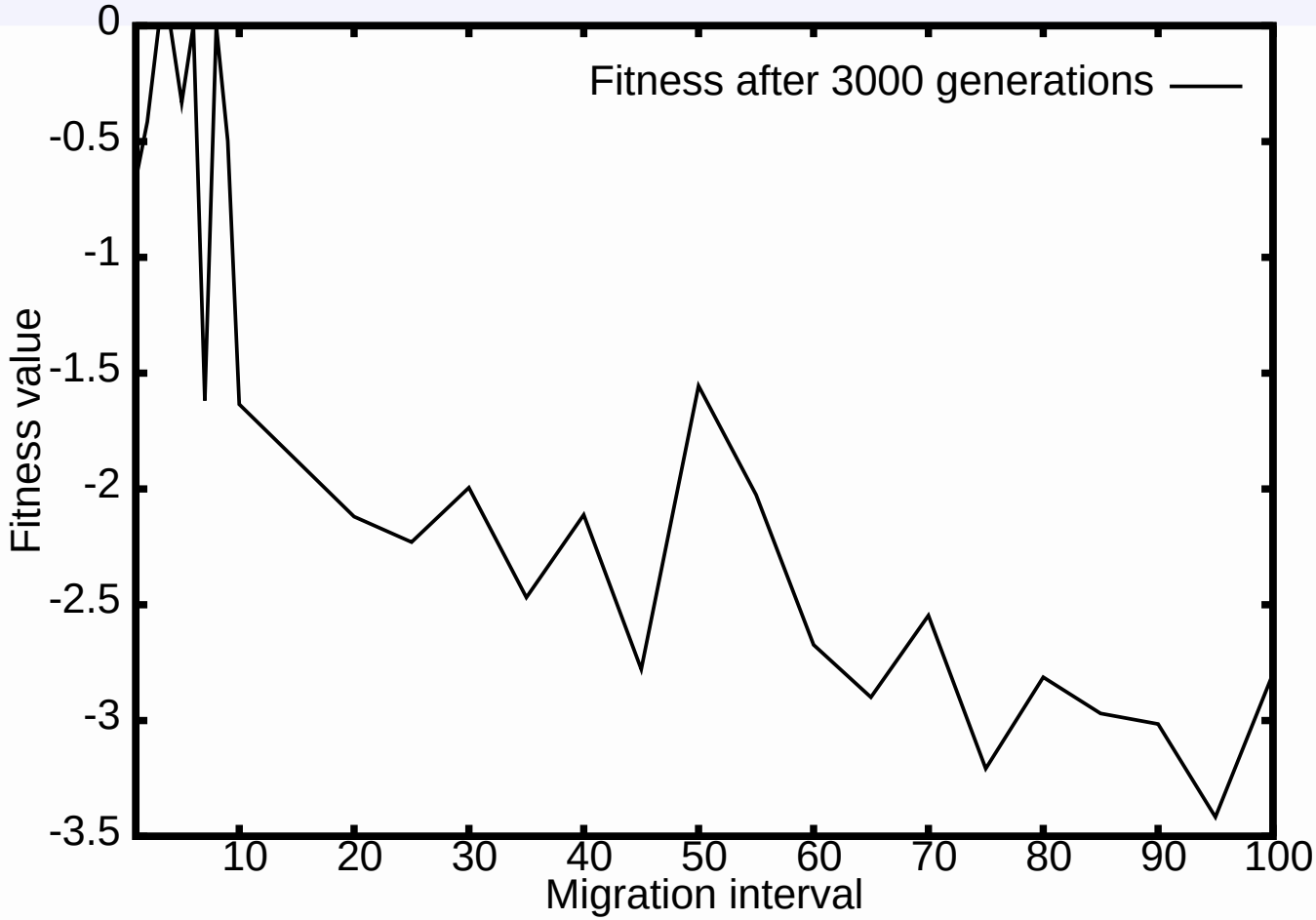
GA and completely random searching

Node scalability



Fitness after 2000 generations

Migration interval



Changing nominal migration interval

Conclusion

- Dynamic Hierarchical System, DNAS
- Implemented Application
  - Genetic Algorithm
- Scalability
  - Scales to about 70 nodes
  - “Migration” was effective