BLUETRONIX SWARM INTELLIGENCE FOR DATA STORAGE/ RETRIEVAL

Motivation

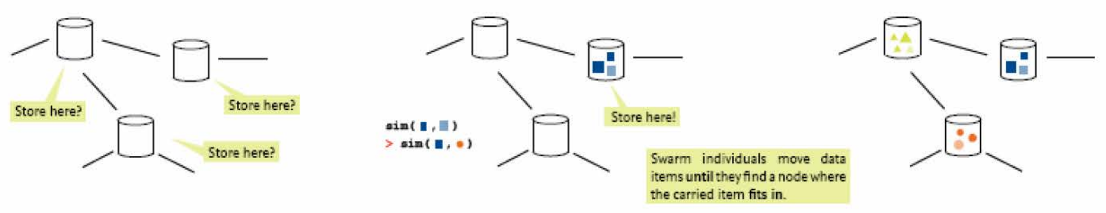
The amount of data handled by semantic applications is expected to increase over a level manageable by available storage systems. Distributed semantic storage solutions are a powerful concept to increase storage capacity. We are in the process of developing a
Self-Organized Semantic Storage Service (S4) using swarm intelligence to overcome present limitations in storage capacity and network dynamics.

Swarm Intelligence

Swarm Intelligence can be described as decentralized behaviour of a large numbers of individuals. Example: **Ant Colonies** perform efficient food retrieval ("Foraging") as well as "Brood Sorting" to organize their offspring.
Swarm algorithms make use of many virtual individuals with limited memory, limited view, and limited lifespan. We have modeled our storage operations to "be" cooperating swarm individuals.

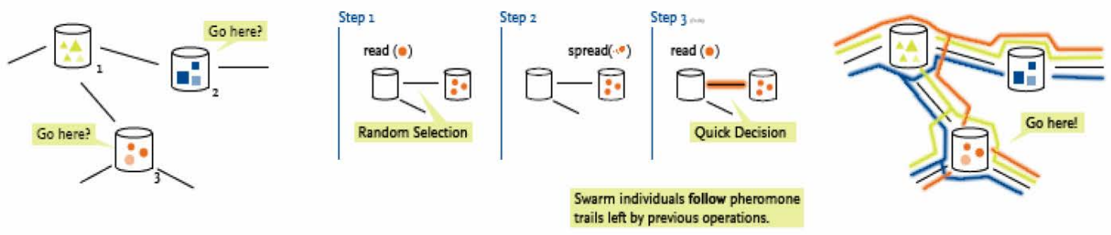
Data Placement and Storage

Task: write (■) - Where should "■" be stored? Swarm Solution: Store **similar data** items on the **same node** Result: Clustered data storage

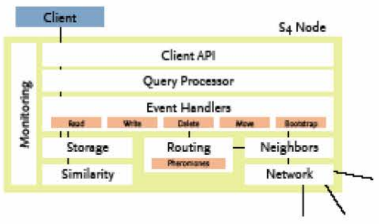

Store here?
Store here?
Store here?
sim(■ , ■)
> sim(■ , ●)
Store here!
Swarm individuals move data items **until** they find a node where the carried item **fits** in.

Data Lookup and Retrieval

Task: read (●) on node 1 - How to find "●"? Swarm Solution: Use **virtual pheromones** to mark paths Result: Pheromone trails facilitate lookup



System Architecture



RDF Storage and Retrieval

RDF Triples are stored in the S4 system by using each triple element as a storage "key" for virtual swarm individuals:

```
store(S,P,O) -> write(S,(P,O)),
write(P,(S,O)), write(O,(S,P))
```

SPARQL queries are evaluated by determining the basic graph patterns within the query and dispatching an individual for each defined component:

```
SELECT ?r WHERE {?r o:p1 'aVal'} ->
  read(o:p1); read('aVal');
```

Similarity Metrics

Task: Define "similarity" for RDF resources:
1) **Syntactic** similarity metric
+ Easy to compute
- Loss of semantic information

```
sim(<p:foo/1>,<p:foo/2>) >
sim(<p:foo/1>,<p:bar/5>)
```

2) **Semantic** similarity metric
+ Chance to exploit RDF structures
- Ontology has to be known

```
sim('cat','dog') > sim('cat','car')
```

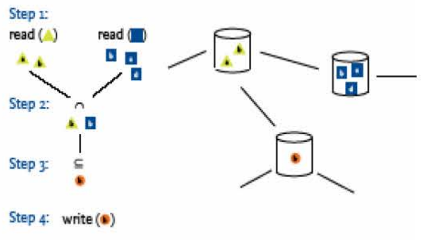
Distributed Reasoning

S4 supports best-effort forward-chaining reasoning: For every axiom added to the TBox, a reasoning individual is spawned. Axioms are matched to assertions using basic lookup operations. Inferred assertions are written back to the store, making them available for retrieval. The reasoning process is subject to the influence of the probabilistic behavior inherent in swarm algorithms. Due to constant change of the stored triples reasoning is also a continuous process. Thus, inferral of new assertions can take some time, and no guarantees can be given whether inferred axioms are retrievable at a certain time.

As an **example**, consider the following axiom:

```
professor ∩ researcher ⊆ seniorresearcher
```

To generate the inferred statements, we have to compute the intersection of all instances of the two concepts, this is achieved by retrieving instances of both concepts using the retrieval mechanism. Their intersection can then be calculated, and the inferred statements are generated and written to the store.



Voice (440) 247.3434 PO Box 23054 Chagrin Falls, Ohio 44023 Fax 440.543.2955

innovation@bluetronix.net www.bluetronix.net