Paramecium: Assembling Raw Nodes Into Composite Cells [⋆]

Ming Chen, Guangwen Yang, Yongwei Wu, Xuezheng Liu
cm01@mails.tsinghua.edu.cn, ygw@mail.tsinghua.edu.cn,
wuyw@tsinghua.edu.cn, xuezhengliu00@mails.tsinghua.edu.cn

Dept. of Computer Science and Technology, Tsinghua University

Abstract. In conventional DHTs, each node is assigned an exclusive slice of identifier space. Simple it is, such arrangement may be rough. In this paper we propose a generic component structure: several independent nodes constitute a cell; a slice of identifier space is under nodes' condominium; part of nodes in the same cell cooperatively and transparently shield the internal dynamism and structure of the cell from outsiders; this type of structure can be recursively repeated. Cells act like raw nodes in conventional DHTs and cell components can be used as bricks to construct any DHT-like systems. This approach provides encapsulation, scalable hierarchy, and enhanced security with bare incurred complexity.

1 Introduction

Many Distributed Hash Tables (DHTs) ([1], [2], [3], [4], [5], [6]) have been proposed in recent years. A distinguishing feature provided by these systems is scalable routing performance while keeping a scalable routing table size in each node. Of those algorithms, every node has a unique numerical identifier and the identifier space is allocated among participant living nodes, which is solely responsible for its assigned exclusive slice, or zone, of identifier space. There is no central node and every node is identical and *visible* to all other nodes. The methodology discarding role difference is simple but may be rigid in practice.

In previous decade, success in object-oriented programming revealed the power of encapsulation: objects encapsulate internal implementations and states; they supersede discrete functions and variables, and communicate through exported public properties and methods. We are enlightened to present Paramecium architecture: raw nodes aggregate into composite cells, which communicate with each other through exported constituent nodes. The extension is simple but promising in two traits: encapsulation and shared zone.

[⋆] This project is supported by the National Natural Science Foundation of China under Grant No. 60373004, No. 60373005, and No. 60273007, and by the National High Technology Development Program of China under Grant No. 2002AA104580

2 Design of Paramecium

Paramecium's goal is to shield highly dynamic behaviors of *unstable* nodes in system and improve security in P2P's open environment. To achieve this goal, Paramecium brings in encapsulation and condominium through cell structure. In this section, we describe Paramecium's cell structure, necessary exported properties and functions. To be generic, we only depict abstract implementation of Paramecium, and leave specific-related issues to concrete implementation. For conciseness, the difference and comparison between Paramecium and other conventional DHTs are here emphasis.

2.1 Cell Structure

Atom Cell Adjacent independent nodes in identifier space constitute an atom cell, which could be identified by an exclusive slice, named *cell zone*, of identifier space. The set of all cell zones covers the whole identifier space. A node resides and only resides in one atom cell. And a cell can be made up of only one node. There is no existent dissociative node. Nodes in the same atom cell are called *sibling nodes*.

Sibling nodes can be organized into flat structure (eg: full connection or DHTs), or hierarchic structure (eg: spanning tree). Paramecium does not specify any material internal structure, including cell zone's division among sibling nodes, and maintenance mechanism in a cell.

Nodes in a cell are categorized into two role types: *boundary nodes* and *hidden nodes*. As representatives of their resident cells, boundary nodes are responsible for requests from nodes in different cells. Hidden nodes facilitate sibling boundary nodes to perform exported functions of resident cell, but they don't serve as representatives. Besides a certain type of request, hidden nodes will reject any request from outsiders. Hidden nodes can directly request non-sibling boundary nodes for services, or sibling boundary nodes for relaying requests. Every node, whether boundary node or hidden node, provides a type of service called *giveMeRepresentatives*. The semantic of *giveMeRepresentatives* is self-explaining and straightforward: when a node X receives a request of this type, X corresponds with the representatives of its resident cell. Role's selection depends on node's discretion and *giveMeRepresentatives*'s implementation. The latter is specific-related and out of the concern of Paramecium. Drawing an analogy between Paramecium and class, we can image that similar to a class, boundary nodes are cell's public methods while hidden nodes are its protected or private methods. Boundary nodes encapsulate implementation of resident cells and export corresponding properties and functions. Hidden nodes' rejection to outer requests enforces the rule of encapsulation. We suggest that representatives free hidden nodes from inter-cell level business, allowing them to concentrate on internal affair. To be concise, we use cell and cell's boundary nodes exchangeably in the rest of this paper if no confusion exists.

Evolve to Organism As a natural extension, the cell structure can be recursively repeated: cells conglomerate into a larger and higher level cell (organism). The grammar expressed in BNF (Backus-Naur Form) is: $cell ::= cell|\{cell\}$. All cells immediately forming a new cell X are called *$X's$ child cell.* This is a hierarchic architecture abiding with the principle of encapsulation. Child cells can serve in two ways: as boundary cells or as hidden cells. Only boundary cells are exported to the outside world of their resident cell. The implementation, maintenance, and internal dynamics of a cell are shielded by its boundary cells, similar to an atom cell. A higher level cell has no knowledge and interest of low level businesses. Considering the consequential benefit, the incurred complexity should be justifiable.

2.2 Modification to Conventional DHTs

Routing Table In addition to its own zone and traditional routing table called inter-cell routing table here in Paramecium, a node must maintain the state composed of its resident cell zone and its *intra-cell routing table.* Each entry in the intra-cell routing table contains NodeID, zone, and other implementation-related information of a sibling, node or subcell. A node's intra-cell routing table can include partial or entire siblings. The connection topology in a cell and the maintenance of intra-cell routing table depend on concrete implementation.

The inter-cell table can be constructed and maintained by Chord[2], Pastry[3], or other DHT protocols. Although an entry in an inter-cell routing table points to an appropriate top-layer cell, the content is cell's boundary node, as well as cell's zone. The selection of boundary nodes through service *giveMeRepresentatives* is also implementation-related. But we must remember that all sibling boundary nodes are eligible candidates. Sibling nodes/subcells can *share* a completely same inter-routing table. Thus, only part of nodes have to actively maintain inter-routing tables and disseminate results to their siblings. This difference tells Paramecium from other DHTs.

Routing The amendment to conventional routing schemas is trivial. A node first checks whether the routing target falls into its resident cell's zone. If yes, the node employs implementation-related approach with the help of intra-cell routing table to route the request (the simplest scenario is that when sibling nodes are full-connected, the final target can be reached in one-hop by local lookup). Otherwise, the request is routed by inter-cell routing table and specific inter-cell routing algorithm.

Node join The join operation is intuitive. Assumed a node X wants to join an existing Paramecium system, X first finds an atom cell C whose zone covers X through routing algorithm described in above paragraph. X then informs other sibling nodes residing in C of joining message. Meanwhile, X learns intra-cell and inter-cell routing tables from them. If X only acts as hidden node, there is no perceivable changes to other cells. Otherwise, other cells will eventually detect the X's arrival in process of periodical update of inter-cell routing table by the service *giveMeRepresentatives.*

Node departure Ordinarily, a node can crash or leave system unpredictably to relevant nodes. Similar to node's join, the departure of a hidden node does not affect other cells' routing table except the node's sibling nodes' intra-cell routing table. The effectiveness of encapsulation apply here again.

3 Related Work

There are many existing or under development DHTs. To the best as we know, Paramecium is the first general architecture that introduces the concepts of encapsulation and jointly governed zone by a group of nodes.

There are some similarities between Paramecium and CAN[1], Chord[2], Pastry[3], Tapestry[4], SkipNet[5], and Koorde[6]. In these conventional DHTs, however, each node exclusively takes portion of identifier space and exposes itself to others in a system level. Nodes can share partial routing table, but they don't support encapsulation, too.

4 Conclusion

Recognizing that raw nodes acting as bricks to build DHTs may not be flexible, Paramecium extends the construction unit from primitive node to composite cell which is made of nodes/subcells in a intuitive and efficient way. The cell structure shields its internal dynamism and structure through the distinguished characteristics of encapsulation: cells interact with each other only through boundary nodes/subcells. Recursive cell composition provides analogy to hierarchal structure in a scalable way. With the help of the trait of jointly shared zone among sibling nodes/subcells, Paramecium can also enhance security to some degree through Practical Byzantine-like algorithms.

References

1. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A Scalable Content-Addressable Network.* In ICSI Technical Report, Jan. 2001.
2. I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan. *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications.* In Proceedings ACM Sigcomm 2001, San Diego, CA, Aug. 2001.
3. A. Druschel and P. Rowstron. *Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer System.* In proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001).
4. B. Y. Zhao, J. Kubiatowicz, A. D. Joseph. *Tapestry: An Infrastructure for Fault-toleratnt Wide-area Location and Routing.* Technical Report UCB/CSD-01-1141.
5. Nicholas J. A. Harvey, Michael B. Jones, S. Saroiu, M. Theimer and A. Wolman. *SkipNet: A Scalable Overlay Network with Practical Locality Properties.* Proceedings of the USENIX Symposium on Internet Technologies and Systems USITS 2003.
6. Frans Kaashoek and David R. Karger. *Koorde: A Simple Degree-optimal Hash Table.* In Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03).