

An Ontology Based Local Index in P2P Networks

Habib Rostami Jafar Habibi Hassan Abolhassani Mojtaba Amirkhani
Ali Rahnama

Computer Engineering Department Sharif University of Technology
{Habib, Habibi, Abolhassani}@sharif.ir, {Amirkhani, ARahnama}@ce.sharif.ir

Abstract

Peer-to-peer networks (P2P) are beginning to form the infrastructure of future applications. One of the problems with unstructured P2P networks is their scalability limitation. This is also the problem in structured P2P networks that use broadcasting to find objects. These networks can't contain a large number of nodes because of the large amount of traffic that they have to handle. Local indexing is a method to reduce this traffic. But local indexes tend to become large with the growth of the network. Also limiting the size of these indexes causes loss of indexing information. In this paper we introduce a novel ontology based local index which limits the size of the local indexes without losing indexing information. We show that the method can be employed by many P2P networks. We also, as an example, apply this local index on HyperCup overlay network, and show that it reduces the network traffic significantly.

1 Introduction

Peer-to-peer networks (P2P) are growing rapidly and unlike current applications like Gnutella [1] and Morpheus [2] that are an infrastructure for file sharing, are developing into an infrastructure for different applications [6], [26], [7], [23], [24]. With the Advent of semantic web [3] and semantic web services [12], many researchers and industry developers use these protocols in P2P networks. P2P networks are characterized by self organization, symmetric communication and distributed control [17]. This self organizing network automatically rearranges itself to joining and leaving nodes. By symmetric communication it is meant that all nodes are both servers and clients. Distributed control means that they don't have a centralized server or servers [18]. Routing is one of the important problems in P2P networks and divides these networks into two categories: structured and unstructured [13]. Messages in unstructured networks are flooded throughout the entire network but in structured networks the messages are routed so that they

pass only a certain number of hops. The earlier versions of Gnutella [1] were unstructured and Chord [21], CAN [15], SkipNet [11], Coral [8], TOPLUS [10] and HyperCup [19] are examples of structured P2P networks. In unstructured P2P networks and some structured P2P networks like HyperCup [19] which use broadcasting to search, a large volume of traffic is created during the search process.

Local indexing methods were introduced to help the message routing process and help reduce unnecessary traffic caused by broadcasting algorithms [22], [5]. But local indexes tend to become large with the growth of the network because these indexes grow with respect to the objects present in the network. So as more objects are added to the network these indexes grow in size. Also by limiting the size of these indexes our search scope is altered in a way that some object can not be reached anymore by these indexes. In this paper we will introduce an ontology based indexing algorithm which solves the mentioned problems by using a shared ontology as a reference for building indexes. We also apply it on HyperCup to enhance it in terms of overlay network traffic. In [20] a method for ontology based routing is offered that changes the structure of the original HyperCup based on semantic services that the nodes provide. We will compare it with a version of HyperCup which uses our ontology based local indexing method.

The rest of this paper is organized as follows: in section 2 we overview some related works. Section 3 includes our proposed ontology based local index (OLI), its data structure and routing algorithm. In section 4 we overview HyperCup P2P network. In section 5 as an example, we discuss the implementation of OLI on HyperCup and show experimental results. And finally in section 6 we conclude our study.

2 Related work

Using semantic information is a trend in recent papers. In this section we will review some related works.

In a work done by Schlosser et al. [20], they changed HyperCup [19] structure by semantic relation between par-

ticipant nodes. They changed the node joining algorithm, so that nodes that have similar services are placed together in concept clusters. In this paper, the nodes that are grouped together in one concept cluster are close in meaning of contents that they provide but not in the physical underlying topology location¹, this causes problems, real world nodes that are linked together may have a long distance between them in terms of latency and physical traffic [16].

In [25], a semantic-based peer similarity measurement for efficient query routing is used. they build up a global dictionary by using WordNet. They update the dictionary using flooding approach. Also, they used some heuristic rules to detect semantic similarity of nodes in the network and a peer answers queries with the help of semantically relevant peers. The topology awareness problem is also a concern in this network.

In [14], we present a Semantic Addressable Network (SAN) which is a combination of both structured and unstructured networks. We focus on CAN [15] and HyperCup [19]. We chose HyperCup because of its efficient query broadcasting and guarantee of non-redundant broadcasts. We group nodes in communities based on their main domain ontology concept. SAN helps solve the following problems: semantic awareness, unnecessary traffic caused by common broadcast methods, and the lost indexing problem in CAN.

3 Ontology based local index

In this section we propose our ontology based local index (OLI). OLI contains a distributed data structure and a routing algorithm which will be described in the following subsections.

3.1 OLI data structure

The data structure by using a limited amount of memory should be able to give each node the information it needs to answer the question "Should I forward the received message on this link?". Suppose that node u receives message msg from link j that contains the concept c and has to decide whether to forward it over link i . Our data structure is a distributed data structure that creates a k -row index (k is defined as a fixed constant natural number over the entire network²) for every link i in each node. This index shows that for node u what semantic contents can be reached through link i . When u receives a message over link j , if the information in the index shows that probably the semantic content c can be reached through link i , the message is forwarded over link i otherwise the message is not forwarded at all. In the rest of this subsection we will describe OLI:

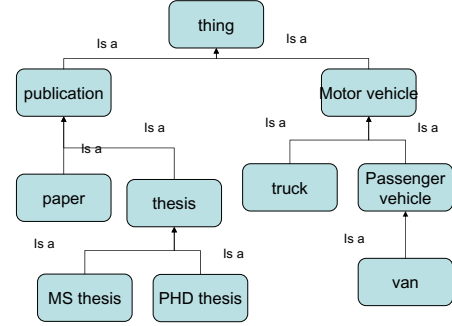


Figure 1. Part of the domain ontology.

Definition 3.1 $Ont(v)$ is a k -row index which each row represents an ontological concept known by the entire network and node v provides contents that can be described by the former concept.

The contents must be represented by just k concepts, so if node v provides more than k semantic contents, it must be summarized to only k concepts that best represent the previous contents.

For example if $k = 2$ and node v provides contents with concepts "paper", "MS thesis" and "PhD thesis" and the hierarchical structure of the ontology is as shown in figure 1. The table will consist of two rows by the concepts of "paper" and "thesis" that "thesis" best represents "MS thesis" and "PhD thesis".

In the rest of this paper we assume that $COntology$ is the object that gives nodes the shared ontology services³.

Definition 3.2 $T_u(i)$ is a k -row table of the concepts of $COntology$ that shows through link i what semantic contents can be accessed by node u . If u by link i is connected to node v then

$$T_u(i) = COntology.aggOnt(Ont(v), T_v(1), \dots, T_v(p)).$$

Where $1, \dots, p$ is the subset of outgoing links connected to node v for which if our algorithm was not applied would have been used to forward the message coming from link i . Method $aggOnt()$ on object $COntology$, gets m k -row indexes from the input and gives a k -row index of the ontology as output which these k concepts describes all of the concepts presented by the input indexes in the most precise manner possible. In fact this is done by finding the least common subsumers of the inputs. The least common subsumer (lcs) of a set of concepts is the minimal concept that subsumes all of them. The minimality condition implies there is no other concept that subsumes all the concepts in the set and is less general than lcs [4].

¹In this case we say that the overlay is not topology aware.

² k is optimized according to the network specification and limitations.

³We assume that there is a shared ontology that is accessible by all nodes and $COntology$ is the interface to the shared ontology.

For example suppose that part of the ontology is as figure 1 and each table has 2 rows ($k = 2$). Then if $t_1 = \{Truck, Van\}$ and $t_2 = \{Passenger\ vehicle, Paper\}$ and $t_3 = C_{Ontology}.aggOnt(t_1, t_2)$ then we have $t_3 = \{Paper, Motor\ vehicle\}$. In other words $aggOnt()$ gets input concepts and gives the k concepts that best describes the inputs.

3.2 The routing algorithm

When node u is sending or receiving the message msg that should be forwarded over link i , it first calls $C_{ontology}.isIn(msg, T_u(i))$, if the answer is positive the message is forwarded over link i , otherwise the message is not forwarded at all. The following pseudo code shows the routing algorithm when node u has received message msg over link j .

```

u.OnReceive(msg , j){
  for(i = 1 ; i <= n; ++i){
    if (msg should be forwarded
        according to base algorithm){
      if (C_ontology.
          isIn(msg , T_u(i))){
        send(i , msg);
      }
    }
  }
}

```

The method $isIn$ shows that if the concept contained in msg matches one of the concepts in the local ontology based index of connection i ⁴.

4 HyperCup

HyperCup is a symmetric overlay based on hypercube. It consists of a structure, a broadcast algorithm and a management algorithm for joining and departing nodes. HyperCup by limiting the number of links between nodes presented a broadcast algorithm that with the least number of messages sent guarantees all nodes will receive the message in at most $\log_2 n$ steps.

In the HyperCup overlay, every node should be able to be a root of a spanning tree of the graph. The nodes can be organized in a n -dimensional hypercube. There are $N = 2^{l_{max}+1}$ nodes in a complete hypercube graph that l_{max} is the dimension of the graph. Edges in the graph are labelled: Node Y is dubbed i -neighbor of node X ($Y = iN(X)$) iff node Y is X 's neighbor in dimension i . Node Y is also referred to X 's neighbor on neighbor level i . Edges in the graph are undirected. A node can have extended neighbors $Y = N(X) = \{x_0, x_1, \dots\}(X)$, where N is termed neighbor link set, and it denotes the sequence

⁴If the message carries concept c then we say that c matches concept r if c equals r or one of the sub-concepts of r .

of i -neighbors one would have to follow in the complete hypercube graph to reach node Y from node X and vice versa (since the i -neighbors are unique). It should be noted that when the hypercube is not complete, for example we have 5 nodes, the hypercube constructing algorithm adds some virtual nodes and creates a complete hypercube. So in the search algorithm we always suppose that the hypercube is complete [19].

HyperCup guarantees that the set of nodes traversed strictly increases during a forwarding process, i.e. nodes receive a message exactly once. It guarantees that exactly $N - 1$ messages are required to reach all nodes in the overlay. Furthermore, the last nodes are reached after \log_2^N forwarding steps. The algorithm works as follows: A node invoking a broadcast sends the broadcast message to all its neighbors, tagging it with the edge label on which the message was sent. Nodes receiving the message restrict the forwarding of the message to those links tagged with higher edge labels [19].

5 Implementation on HyperCup

In this section we discuss the implementation of OLI on HyperCup. Suppose that our HyperCup has $l_{max} + 1$ dimensions, so the links are labelled $0, \dots, l_{max}$. In this version of HyperCup if u by link i is connected to node v then $T_u(i) = C_{ontology}.aggOnt(Ont(v), T_v(i + 1), \dots, T_v(l_{max}))$.

When node u is sending or receiving the message msg that should be forwarded according to the HyperCup routing algorithm over link i , first it calls $C_{ontology}.isIn(msg, T_u(i))$, if the answer is positive the message is forwarded over link i (supposed that the message is created by u or has been received over link $j < i$, and according to the HyperCup routing algorithm has to be forwarded) otherwise the message is not forwarded. The following pseudo code shows the routing algorithm when node u has received message msg over link j .

```

u.OnReceive(msg , j){
  for(i = j + 1 ; i <= l_max; ++i){
    if (C_ontology.isIn(msg , T_u(i))){
      send(i , msg);
    }
  }
}

```

u uses the method $isIn$ for only those links that are labelled greater than j . This is because in the HyperCup algorithm when a node receives a message over link j it can only forward it over links with labels greater than j .

5.1 Join and departure of nodes in the network

The important problem that has to be considered is that the information in some indexes may not be valid when a node joins or departs the network or when a node provides

a new content or discontinues an old content. So when one of the following occurs the indexes have to be updated: (1) a node joins the network (2) a node leaves the network (3) adding or stopping or changing semantic descriptions of contents of a node.

We define a special message *updateMsg* which its broadcast throughout the network is not the same as regular messages. When node *v* receives message *updateMsg* over link *i* the routing algorithm for this special message is as follows: First, method *v.updateMsg()* is called and then in contrast to the method in HyperCup the message *updateMsg* is forwarded over all links *j* that *j < i*. The composer node of the message *updateMsg* forwards this message over all of its links like regular messages. The method *v.update()* is as follows.

```
v.update(i) {
    T_v(i) = COntology.aggOnt(Ont(u),
        T_u(i + 1), ...,
        T_u(l_max)
    );
}
```

The following pseudo code is used when node *v* has received message *updateMsg* over link *i*.

```
v.OnReceiveUpdateMsg(i) {
    v.update(i);
    for( j = 0; j < i; ++j){
        send(j, updateMsg);
    }
}
```

When node *u* joins the network, first for every connection *i* that is established with other nodes the index $T_u(i)$ is built and then forwards message *updateMsg* over all the links connected to it.

When a neighbor of node *u* with link *i* to *u* leaves the network, node *u* sends the message *updateMsg* over all of its links that have labels less than *i*.

Finally when a node *u* changes its content (adding or stopping or changing contents semantic descriptions), after updating $Ont(u)$, transmits the message *updateMsg* over all its links.

5.2 Experimental results

In this subsection we discuss experimental results of implementing the algorithm on HyperCup network. We extended PlanetSim simulator [9] to do experiment.

Table 1 represents the simulation parameters. We fix size of the overlay network and examine other parameters. Figures 2 and 3 show the effect of changing the size of ontology table of each link ($k = T_i(u)$) on traffic improvement. As figure 2 shows, even small size ontology tables (i.e. 3) can reduce overall traffic (search traffic, OLI overhead and churn overhead) and search traffic by about 30 percent, which is a great improvement.

Name	Meaning
SQ	Percent of nodes including the searched concept
K	Size of ontology table over each link
N	Average number of concepts that each node contains
TN	Size of ontology table of each node
P	Average queries per join,leave or changing concepts of a node
Q	Average number of concepts each search query includes
Q_OR	Percent of search queries with or between parameters
Q_AND	Percent of search queries with and between parameters
S	Network size

Table 1. Simulation parameters.

On the other side, if we choose the size of tables very large (i.e. 28 and more) we just gain an improvement about 50 percent. Therefore in practical cases we should choose small size ontology tables (for sake of saving nodes memories) and gain a significant improvement in overall traffic as well as search traffic.

OLI also reduces unnecessary messages, messages that are created without leading to an answer node, significantly (figure 3). It means that OLI effectively prune branches of the search trees which not contain a proper answer.

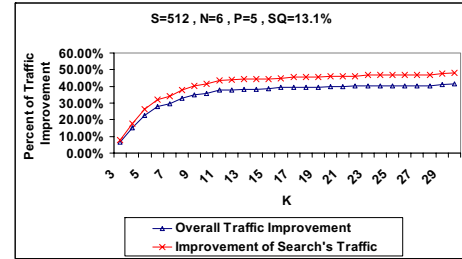


Figure 2. Effect of size of ontology table over each link on traffic improvement.

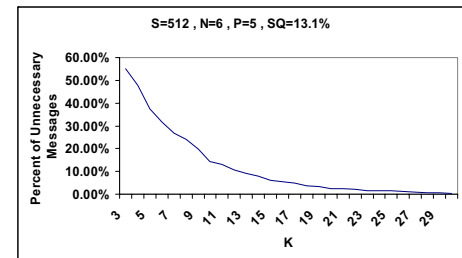


Figure 3. Effect of size of ontology table over each link on reduction of unnecessary messages.

Figures 4 and 5 show that OLI is very effective when we search a concept that a few percent of nodes, host it. If in average, each concept is hosted by about 3 percent of the network nodes, then OLI improves the overall traffic and search traffic about 50 percent (when $s = 512$, $n = 6$, $k = 12$ and $p = 5$) (see figure 4). So in real world we expect a significant improvement by OLI. Also figure 5 shows that the algorithm overhead gradually decreases in comparison with overall traffic when nodes contains popular concepts. Therefore OLI is effective both when the network contains popular concepts or sparse ones.

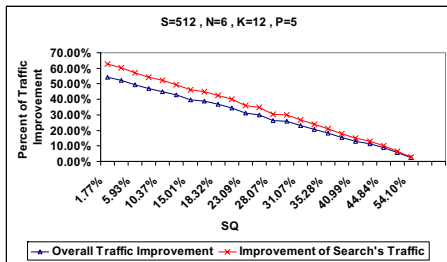


Figure 4. Effect of percent of nodes including the searched concept on traffic improvement.

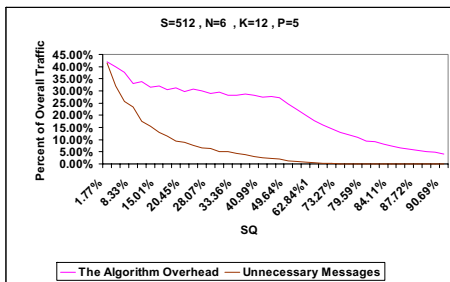


Figure 5. Effect of percent of nodes including the searched concept on reduction of unnecessary messages.

Figure 6 shows relation of Average queries per churn rate (or rate of changing concepts of nodes) with improvement of overall traffic of network as well as search queries traffic. It can be seen that OLI is more effective when the network is stable and churn rate is low. When the network is stable OLI can improve the overall traffic as well as search traffic by more than 60 percent.

Figure 7 shows relation of OLI effectiveness with size of ontology table of each node. As we mentioned before, each node maintains a table per link as well as a table for the node itself. As it can be seen, if we choose the table large enough, then effectiveness of OLI will be increased.

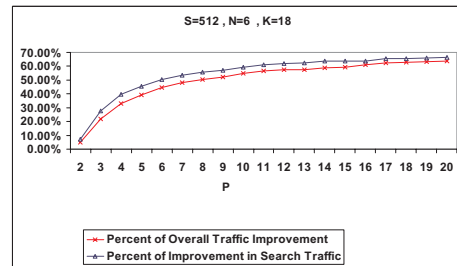


Figure 6. Effect of average queries per join,leave or changing concepts of a node on traffic improvement.

It suggests us that in practice, each node should allocate good enough memory to the ontology table.

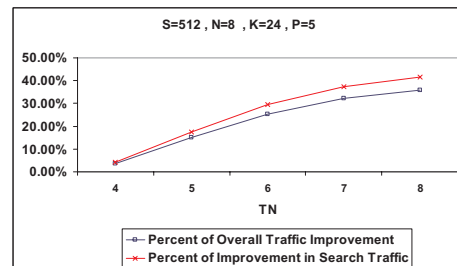


Figure 7. Effect of size of ontology table of each node on traffic improvement.

A user can send a query that contains some concepts with OR/AND operators between concepts. Figure 8 shows that OLI is more effective when number of queries contain AND operators is more than queries contain OR operators. This phenomena is expected because when the query contains AND operators, OLI only allows the query proceed over links that their ontology table address all concepts.

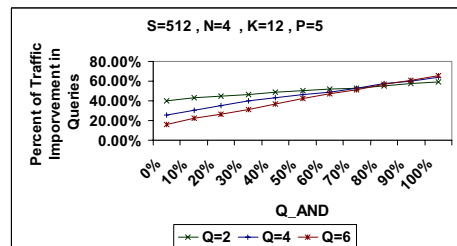


Figure 8. Effect of percent of queries contain and operators on efficiency of OLI.

As experimental results shows, OLI effectively reduces traffic cost of HyperCup P2P network and for better per-

formance, it should be configured. Based on experimental results we expect that in real world this technique can cause P2P networks more scalable, because traffic overhead of P2P networks is the main bottleneck of scalability of these networks.

6 Conclusion and future work

In this paper we introduced an ontology based indexing method (OLI) which, in addition to having the advantages of traditional indexing methods, doesn't have their disadvantages. We showed that OLI is a general method that can be implemented on many P2P networks and can be a base for future developments in this area. And as an example we simulated OLI on HyperCup P2P network and showed that it significantly improves the network traffic (in some configurations, it reduces the network traffic by more than 60 percent). As future work we intend to implement OLI on some other structured and unstructured P2P networks and measure its influence on their performance.

References

- [1] Gnutella website. <http://www.gnutella.com>.
- [2] Morpheus website. <http://www.musiccity.com>.
- [3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May 2001.
- [4] W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumer in description logics. In W. Swartout, editor, *Proc. 10th Nat Conference on Artificial Intelligence (AAAI92'92)*, pages 754–760, San Jose, CA, 1992. MIT, MIT Press.
- [5] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. 28th Conference on Distributed Computing Systems*, July 2002.
- [6] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15:200–222, March/April 2001.
- [7] G. Fox, S. Pallickara, and X. Rao. A scaleable event infrastructure for peer-to-peer grids. In *JGI '02: Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande*, pages 66–75, New York, NY, USA, 2002. ACM Press.
- [8] M. Freedman and D. Mazieres. Sloppy hashing and self-organizing clusters. In *Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, March 2003.
- [9] P. Garca, C. Pairot, R. Mondjar, J. Pujol, H. Tejedor, and R. Rallo. Planetsim: A new overlay network simulation framework. In *Software Engineering and Middleware (SEM 2004)*, Springer LNCS, volume 3437, pages 123–137, 2005.
- [10] L. Garces-Erice, K. Ross, E. Biersack, P. Felber, and G. Urvoy-Keller. Topology-centric lookup service. In *Proc. The 5th International Workshop on Networked Group Communications (NGC'03)*, 2003.
- [11] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proc. The Fourth USENIX Symposium on Internet Technologies and Systems (USITS03)*, March 2003.
- [12] McIlraith, S. Son, and T. Zeng. Semantic web services. *IEEE Intelligent Systems, Special Issue on the Semantic Web*, 16:46–53, March/April 2001.
- [13] J. Mishchke and B. Stiller. A methodology for the design of distributed search in p2p middleware. *IEEE Network*, 18(1):30–37, 2004.
- [14] A. Rahnama, J. Habibi, H. Rostami, and H. Abolhassani. A semantic addressable network. In *IADIS International Conference WWW/Internet*, pages 43–51, 2005.
- [15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. Scalable content-addressable networks. In *Proc. ACM SIGCOMM'01*, 2001.
- [16] H. Rostami and J. Habibi. A mathematical foundation for topology awareness of p2p overlay networks. In *the 4th International Conference on Grid and Cooperative Computing (GCC2005)*, Springer LNCS, volume 3795, pages 906–918, 2005.
- [17] M. Roussopoulos, M. Baker, D. Rosenthal, T. Guili, P. Maniatis, and J. Mogul. 2 p2p of not 2 p2p. In *The 3rd International Workshop on Peer-to-Peer Systems*, San Diego, CA, USA.
- [18] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *International conference on Distributed Systems platforms (Middleware)*, pages 329–350, 2001.
- [19] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. Hypercup shaping up peer-to-peer networks. Technical report, Stanford University, 2001.
- [20] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. Hypercup - hypercubes, ontologies and efficient search on p2p networks. In *International Workshop on Agents and Peer-to-Peer Computing*, Bologna, Italy, July 2002.
- [21] I. Stoica et al. Chord: A scalable p2p lookup service for internet applications. In *Proc. ACM SIGCOMM'01*, 2001.
- [22] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proc. ICDCS*, 2002.
- [23] H. Zhuge. China's e-science knowledge grid environment. *IEEE Intelligent System*, 19(1):13–17, 2004.
- [24] H. Zhuge. *The Knowledge Grid*. World Scientific, 2004.
- [25] H. Zhuge, J. Liu, L. Feng, X. Sun, and C. He. Query routing in a peer-to-peer semantic link network. *Computational Intelligence*, 21(2):197–216, 2005.
- [26] H. Zhuge, X. Sun, J. Liu, E. Yao, and X. Chen. A scalable p2p platform for the knowledge grid. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1721–1736, December 2005.