www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 2 February, 2014 Page No. 3897-3904

CACHING DATA THROUGH PULL BASED APPROCHES IN WIRELESS NETWORKS

S. Mahalakshmi, Dr. M. Kamarajan

Abstract-In MANET environments, data caching are essential because it increases the ability of mobile to access desired data, and improve overall system performance. This paper proposes distributed cache invalidation mechanism (DCIM), a client based cache consistency scheme that's implemented on prime of an antecedently planned design for caching information things in mobile impromptu networks (MANET),namely COACS, where special nodes cache the queries and therefore the addresses of the nodes that store the responses to those queries. DCIM may be a pull-based algorithmic program that implements adaptive time to measure (TTL),piggybacking, per-fetching and provides near study consistency capabilities. Cached knowledge things as appointed adjective TTL values that correspond to their update rates at the data supply, where ever things with terminated TTL values as stored in validation requests to the information adjective to refresh them, where a sun expired ones however with high request rates as perfected from the server. DCIM is analyzed to assess the delay and bandwidth gains in comparison to polling whenever and push based schemes.

Index Terms- Cache invalidation, client polling, consistency data, pull based scheme, TTL

INTRODUCTION

Ad hoc is a Latin phrase meaning "for this". It generally signifies a solution designed for a specific problem or task, non-generalizable, and not intended to be able to be adapted to other purposes (compare *a* priori). Common examples are organizations, committees, and commissions created at the national or international level for a specific task. In other fields the term may refer, for example, to a military unit created under special circumstances, a tailor-made suit, a handcrafted network protocol, or a purpose-specific equation. Ad hocking also mean makeshift solutions, shifting contexts to create new meanings, inadequate planning, or improvised events.

A wireless ad hoc network is a decentralized type of wireless network. The network is ad hoc because it does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically on the basis of network connectivity. In addition to the classic routing, ad hoc networks can use flooding for forwarding data. An ad hoc network typically refers to any set of networks where all devices have equal status on a network and are free to associate with any other ad hoc network device in link range.

Ad Hoc network is a self-organizing multi-hop wireless network, which relies neither on fixed infrastructure nor on predetermined connectivity.

Ad Hoc networks can be classified using various parameters:Symmetric and Asymmetric, Traffic Characteristics, Routing Methods, Some other metrics such as time and reliability constraint

Driving means changing constantly location. This means a constant demand for information on the current location and specifically for data on the surrounding traffic, routes and much more. This information can be grouped together in several categories.

A very important category is driver assistance and car safety. This includes many different things mostly based on sensor data from other cars. One could think of brake warning sent from preceding car, tailgate and collision warning, information about road condition and maintenance, detailed regional weather forecast, premonition of traffic jams, caution to an accident behind the next bend, detailed information about an accident for the rescue team and many other things. One could also think of local updates of the cars navigation systems or an assistant that helps to follow a friend’s car. Another category is infotainment for passengers. For example internet access, chatting and interactive games between cars close to each other. The kids will love it.

Next category is local information as next free parking space (perhaps with a reservation system), detailed information about fuel prices and services offered by the next service station or just tourist information about sights .A possible other category is car maintenance. For example online help from your car mechanic when your car breaks down or just simply service information. The decentralized nature of wireless ad hoc networks makes them suitable for a variety of applications where central nodes can't be relied on and may improve the scalability of networks compared to wireless managed networks, though theoretical and overall capacity of such networks have been identified.

Minimal configuration and quick deployment make ad hoc networks suitable for emergency situations like natural disasters or military conflicts. The presence of dynamic and adaptive routing protocols enables ad hoc networks to be formed quickly.

1.Secure Data Transmission in Mobile Ad Hoc Networks

The vision of nomadic computing with its ubiquitous access has stimulated much interest in the Mobile Ad Hoc Networking(MANET) technology. However, its proliferation strongly depends on the availability of security provisions, among other factors. In the open, collaborative MANET environment practically any node can maliciously or selfishly disrupt and deny communication of other nodes. In this paper, we present and evaluate the Secure Message Transmission (SMT) protocol, which safeguards the data transmission against arbitrary malicious behavior of other nodes.

SMT is a lightweight, yet very effective, protocol that can operate solely in an end-to-end manner. It exploits the redundancy of multipath routing and adapts its operation to remain efficient and effective even in highly adverse environments. SMT is capable of delivering up to 250% more data messages than a protocol that does not secure the data transmission. Moreover, SMT outperforms an alternative single-path protocol, a secure data forwarding protocol we term Secure Single Path (SSP) protocol. SMT imposes up to68% less routing overhead than SSP, delivers up to 22% more data packets and achieves end-to-end delays that are up to 94% lower than those of SSP. Thus, SMT is better suited to support Quos for real-time communications in the ad hoc networking environment. The security of data transmission is achieved without restrictive assumptions on the network nodes’ trust and network membership, without the use of intrusion detection schemes, and at the expense of moderate multi-path transmission overhead only.

The communication in mobile ad hoc networks comprises two phases, the route discovery and the data transmission. In an adverse environment, both phases are vulnerable to a variety of attacks. First, adversaries can disrupt the route discovery by impersonating the destination, by responding with stale or corrupted routing information, or by disseminating forged control traffic. This way, attackers can obstruct the propagation of legitimate route control traffic and adversely influence the topological knowledge of benign nodes. However, adversaries can also disrupt the data transmission phase and, thus, incur significant data loss by tampering with, fraudulently redirecting, or even dropping data traffic or injecting forged data packets.

2.Defending Against Cache Consistency Attacks in Wireless Ad Hoc Networks

Caching techniques can be used to reduce bandwidth consumption and data access delay in wireless ad hoc networks. When cache is used, cache consistency issues must be addressed. To maintain strong cache consistency in some strategic scenarios (e.g., battle fields), the invalidation based approach is preferred due to its low overhead. However, this approach may suffer from some security attacks. For example, a malicious node (intruder) may drop, insert or modify invalidation messages to mislead the receivers to use stale data or unnecessarily invalidate the data that is still valid. In this paper, we propose a solution based on the IR-based cache invalidation strategy to prevent intruders from dropping or modifying the invalidation messages .Although digital signatures can be used to protect IRs, it has significantly high overhead in terms of computation and bandwidth consumption. To address this problem, we propose a family of randomized grouping based schemes for intrusion detection and damage recovery. Extensive analysis and simulations are used to evaluate the proposed schemes. The results show that our solution can achieve a good level of security with low overhead.

In wireless ad hoc networks, nodes communicate with each other using multi-hop wireless links. Due to lack of infrastructure support, each node acts as a router, forwarding data packets for other nodes. Most of the previous research in ad hoc networks focuses on the development of dynamic routing protocols that can efficiently find routes between two communicating nodes. Although routing is an important issue in ad hoc networks, other issues such as information(data) access are also very important since the ultimate goal of using ad hoc networks is to provide information access to mobile nodes.

3.A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments

The falling cost of both communication and mobile terminals (laptop computers, personal digital assistants, hand-held computers, etc.) has made mobile computing commercially affordable to both business users and private consumers. In the near future, people with battery powered mobile terminals (MTs) can access various kinds of services over wireless networks at any time or any place. However, due to limitations on battery technologies these MTs may be frequently disconnected (i.e., powered off) to conserve battery energy. Also, the wireless Bandwidth is rather limited.

Caching frequently accessed data on the client side is an effective technique for improving performance in a mobile environment. Average data access latency is reduced as several data access requests can be satisfied from the local cache, thereby obviating the need for data transmission over the scarce wireless links. However, frequent disconnections and mobility of the clients make cache consistency a challenging problem. Effective cache invalidation strategies are required to ensure the consistency between the cached data at the clients and the original data stored at the server.

When cache techniques are used, data consistency issues must be addressed to ensure that clients see only valid states of the data, or at least do not unknowingly access data that is stale according to the rules of the consistency model. Problems related to cache consistency have been widely studied in many other systems such as multiprocessor architectures distributed file systems distributed shared memory and client-server database systems. Depending on whether or not the server maintains the state of the client’s cache, two invalidation strategies aroused: the shameful server approach and the stateless server approach.

In the shameful server approach, the server maintains the information about which data are coached by which client. Once a data item is changed, the server sends invalidation messages to the clients with copies of the particular data. The Andrew File System is an example of this approach. However, in mobile environments, the Server may not be able to contact the disconnected clients.

Thus, a disconnection by a client automatically means that its cache is no longer valid. Moreover, if the client moves to another cell, it has to notify the server. This implies some restrictions on the freedom of the clients. In the stateless server approach, the server is not aware of the state of the client’s cache. The clients need to query the server to verify the validity of their caches before each use. The Network File System (NFS) is an example of this approach. Obviously, in this option, the clients generate a large amount of traffic on the wireless channel, which not only wastes the scarce wireless bandwidth, but also consumes a lot of battery energy.

4.Data Consistency for Cooperative Caching in Mobile Environments

He trend toward wireless communications and advances in mobile technologies are increasing Consumer demand for ubiquitous access to Internet-based information and services. However, Due to battery power limitations, users often must disconnect mobile devices from the network to conserve energy. Moreover, wireless links have lower capacity than wired links and wireless channels are less stable, resulting in higher network congestion and packet loss.

These challenges make mobile communication unreliable; emphasizing the need for efficient information-access mechanisms. Cooperative caching improves system performance because it allows sharing and coordination of cached data among multiple mobile users in the network. By cooperatively caching frequently accessed information, mobile devices do not always have to send requests to the data source. In addition to reducing query latency, this technique can lower mobile host communication overhead and energy consumption. However, the unreliable communication and the user’s mobility make it difficult to maintain cache consistency.

As the “Cache Invalidation Techniques” sidebar describes, researchers have proposed a wide range of strategies for maintaining cache consistency in recent years. However, each strategy has its own design goals and application scenarios. No uniform or structured methods exist for current work, making it difficult to evaluate their relative effectiveness and performance. In response to this problem, we have developed a 3Dmodel that captures the main features of cache consistency schemes and provides a basis for evaluating existing strategies as well as designing new ones. Based on this model, we propose a hybrid and generic strategy: relay-peer-based cache consistency. Because RPCC uses relay peers between the source hosts and cache hosts to forward updated information, it can divide cache invalidation into two asynchronous procedures.

Mobile wireless environments can be broadly classified as either infrastructure based or ad hoc based. In the former scheme, a fixed network device such as a mobile support station forwards messages that mobile hosts send or receive.

The MSS is similar to the server in a traditional client-server distributed system in that all source data is deployed on it. Other mobile hosts retrieve data from the MSS and can cache a replica by themselves. In contrast, ad hoc networks, like that shown in Figure 1b, do not store data on the MSS but use it only as the access point to the Internet. Ad hoc networks disperse all data items for searching and querying across the mobile hosts. Thus, the cache invalidation strategies employed in a single-hop wireless mobile network are not suitable for a multichip mobile ad hoc network.

5. An Update-Risk Based Approach to TTL Estimation in Web Caching

As the web grows into an infrastructure for disseminating information, so does the volume of data exchanged on the web. This explosive growth of the data volume is overloading the web servers and the communication network, causing performance degradation. Web caching offers a Solution to this problem by retaining frequently used web pages on the client side recent research outputs in this area have been incorporated into such commercial products.

Like any cached data, cached web pages are copies of the original data (from the web server), and therefore, need to be synchronized with the original data. This consistency requirement is rather weak in the case of web caching as discussed in the references. That is, it allows the synchronization to be delayed, and thus, allows the clients to access outdated data for some time Synchronization delay affects consistency and performance of cache. Larger synchronization delay results in lower consistency, but better performance of cache. In other words, cache consistency and performance have a tradeoff relationship. Therefore, the cache server administrator should find a reasonable compromising point between consistency and performance of cache.

The rest of the paper is organized as follows. We give an overview of the web caching techniques and describe the conventional TTL estimation methods. In Section 3 we explain the concept of the update risk and develop a mechanism for determining the TTL based on the update risk. In Section 4 we describe the experiments performed and present the results. In Section 6 we compare our method with the two conventional methods in terms of the update risk.

Based on their positions in the network, web cache servers are classified into the reverse cache server, the transparent cache server, and the proxy cache server as shown in First, the reverse cache server is placed in front of the web server. It reduces the load on the web server by responding to clients' requests on behalf of the server, thereby helping the server scale up to handle heavy workload. Second, the transparent cache server and the proxy cache server are used for multiple clients to share and reuse the data accessed in the same local area network (LAN).These cache servers offer the following benefits obtained by obviating the accesses to remote servers: reduced response time, reduced network traffic, and reduced load on the web server. To use the proxy cache server, a client should explicitly indicate it in its configuration.

To use the transparent cache server, a client does not have to because all requests ,regardless of the client's configuration, are redirected to the cache server by a network switch. The client cache in simply an internal cache of the individual browsers. Among these cache servers, our focus is on the client-side cache server. Therefore, from now on our scope is confined to the transparent cache server and the proxy cache server.

The fixed TTL method is used in IIS of Microsoft. It Assigns the same TTL to all data items regardless of the time of update, and thus, is simple and easy to implement. However ,the reality is that different data items are updated at different rates, and even the same data item is updated at different rates at different times. Thus, this method lacks any consideration for distinct and time-variant update patterns of data items at all. This deficiency deprives the method of its credibility in setting a reasonable TTL. If the TTL is set too large, the cache server ends up using obsolete data longer than necessary. If the TTL is set too small, it end sup revalidating unnecessarily often, thus wasting the system and network resources.

6.Piggyback Server Invalidation for Proxy Cache Coherency

Caching is widely used in the Web at the browser and proxy level. Previously, we studied piggyback cache validation (PCV), a technique to improve cache coherency and reduce the cache validation traffic between proxy caches and servers. We focused on using the information available in the cache and partitioned resources on the basis of originating server. When a server is contacted again, the proxy client piggybacked a list of cached, but potentially stale, resources obtained from that server for validation.

The server replied with the subset of resources that are no longer valid. PCV yielded stronger cache coherency and reduced costs by using the validations to extend the expiration time for cached resources and reduce the need for GET If-Modified-Since (IMS) requests.

In this work, we combine resource information available to servers with the piggybacking technique to create a mechanism called Piggyback Server Invalidation (PSI). Servers partition the set of resources at a site into *volumes*, either a single site-wide volume or related subsets of resources and maintain version information for each volume. When a server receives a request from a proxy client containing the client's last known version of the volume, it piggybacks a list of volume resources modified since the client-supplied version.

The proxy client invalidates cached entries on the list and can extend the lifetime of entries not on the list. Servers maintain volume, but no proxy-specific information. While the mechanism could be used by browser clients, we focus our discussion on its use by proxy clients, where there are more cached resources.

The aim of the PCV and PSI mechanisms is the same---use piggybacking to eliminate stale entries from a proxy cache while extending the lifetime of valid entries without introducing additional network traffic. However, the mechanisms differ in their use of resource information and piggybacking. The PCV mechanism uses resource information available only to a proxy while PSI can group resources based on access patterns and modification characteristics known only to a server. The PSI mechanism does require changes to existing Web servers for implementation. The PCV mechanism piggybacks a list of resources in the proxy cache for validation, which can cause validation checks for resources that have not changed. The PSI piggybacks a list invalidated resources, which have changed at the server, but may not be coached at the proxy. For both mechanisms, this unused piggybacked information creates additional bandwidth and latency overhead.

In this work, we study the PSI approach, comparing its performance and overhead to the PCV approach and existing cache coherency techniques. The study was carried out using trace driven simulation on two large independent proxy log data sets. Proxy logs do not have the information regarding when resources change on the server and thus the number of invalidations that would be generated between client accesses. In the absence of such end-to-end logs we studied a number of representative server logs to obtain a measure of invalidations that do occur between successive client accesses and used this measure in the simulation.The rest of this paper is organized as follows: we begin with a discussion of related work in the field of file systems and the Web in describes piggyback server invalidation in more detail and discusses approaches for its implementation. Presents the environment and evaluation criteria for studying various PSI-based cache coherency policies. Provides the results of this study. Summarizes the results and discusses alternative approaches.

7.Minimizations of the Update Response Time in a Distributed Database System

The basic architecture of a distributed database system (DDBS) consists of database sites connected to each other via a communication network. At each database site, there is a computer running one or both of the software modules, namely data access software which supervises user interactions with the database, and data storage software which stores and manages the physical data at each site In the case of full replication, an update transaction has to be broadcast from an access site to all the storage sites of the DDBS.

The update transactions to be processed on a given data item may arrive in differing orders at distinct storage sites. The integer timestamp ordering (TO) algorithm, known as the Leann's ticketing algorithm can be used in packet switched networks where messages may arrive out of sequence. The global order is established in the following way the access sites are ordered in a virtual ring configuration. On this ring, *M* tokens circulate (one token for every data item in the database). These tokens are in charge of sequentially delivering the timestamp.

The system analyzed in is a one-stage DDBS as opposed to the two-stage DDBS investigated in this paper. The two-stage DDBS is shown in. It is assumed that the token passing mechanism of Leann's ticketing algorithm is implemented at access sites. It is further assumed that the allocation of timestamps is very fast compared to the arrival of the updates. The propagation topology for the updates involves two stages: the first stage of communication is from the access sites to *K* gateway controllers, and the second stage is from the gateway controllers to the end-users' *L* storage sites.

Proposed system considers fully replicated databases. An update is replicated to produce *K* new updates and is immediately sent to the input queue of the communication link in order to broadcast to all gateway controllers. Update customers that arrive from the communication medium are re sequenced in a re sequencing buffer with respect to the TO, defined among the different access sites by the token mechanism. In the proposed system, it is assumed that a batch server will serve periodically and together the waiting in-sequence update customers. Each gateway controller will then generate new time stamps by maintaining a counter and will issue a new timestamp for every batch.

This newly formed update customer is replicated to produce Elk new customers, which are immediately sent to the input queue of the communication link in order to broadcast to all the end-users' storage sites. At storage sites, servers will update customers according to a "re sequencing" service discipline. Once serviced, all customers leave the queuing network and a "join" takes place when all replicas of a given data item have been updated.

This arrangement enables the re sequencing delay to be controlled at the second stage, to minimize the processing time at end-users and hence reduce the overall update response time. This novel re sequencing strategy is referred to as hop-by-hop re sequencing with batch processing the idea of batch processing was motivated by the fact that the departure process in the re sequencing buffer is a batch departure. The proposed technique is adaptive by nature, since with any changes in network parameters the batch-processing period T can be changed to minimize the end-to-end delay.

8.On the Cooperation of Web Clients And Proxy Caches

Web cache is an indispensable component in the Internet. It reduces response time and saves network bandwidth consumption. These benefits are achieved via two mechanisms: caching and validation. Caching is an age-old technique for reducing access latency in many systems. It avoids new transactions with the remote server for objects that are still fresh. Between a client and a web server, there could be more than one cache; these caches form a chain, and the sites hosting these caches are commonly referred to as proxy servers.

Every cached object has its own lifetime called time-to-live (TTL), where an object within its TTL is considered fresh and will be served from the cache upon request.
Server generates a response; a validated is included in the header. The requesting cache keeps the validates with the object received. A valuator can be considered as the "version number" of the object. When an object expires (according to the cache's own rule) and is then referenced, its Valuator is sent with the request to the next site up in the chain towards the web server. These requests are called validation requests. The receiving site checks the received validates against its own validated of the object. If the validations match, a short Not Modified response will be generated. In this case, the cache will consider its copy of the object still being fresh and update the expiration time of the object. Otherwise, a full response including the new document and it's validate is sent from the receiving site. Describes the sequence of actions of handling an object request.

When a reference to a stale object is made at a cache, a validation request of the object will be sent to a higher level cache if one exists, or to the web server, as described in A Not Modified response, or an OK response with the object will be returned to the requesting cache. If it is a Web server that generates the response, a new Date header is included in the response. If it is a proxy cache that generates the response, the cached Date header will be sent, together With an Age header to indicate how long the object has been cached.

Every cached object has a fixed TTL value which is computed when the object is received. The Date header is used in finding the TTL, which will be discussed in the following subsections. There is also an age associated with each object. The age value is a local estimation of the time that the object has resided in the entire cache system. The age value grows with time. When the age of an object exceeds its TTL, it becomes stale. The Age header in the response tells the requesting cache the responding cache's own estimation on the age of the object, which in general is used by the requesting cache as the age of the object when it is received.

Another unique characteristic of caching in the web is that some objects are uncatchable. Querying the cache for all the objects, as in traditional caching systems, is thus not suitable for web caching. It is a pure overhead to request a cache for uncatchable object and the result is increased response time and system load. It is therefore superfluous for a client to issue requests of uncatchable objects to the proxy cache. Studies on web proxy cache only reported the existence of such problem without actually solving it. Commercial web caching solutions alleviate the problem by employing a web switch that differentiates requests of cacheable objects from the others. Cacheable object requests are sent to the proxy, while the uncatchable ones are directed to the web servers.

All the values without a subscript in the equations above are specified by the origin server in the HTTP response header. Caches do not change these values and hence they are the same in all the caches involved. As a result, Equations1 and 2 will yield the same values in all the caches, and all the copies will expire at the same time if the clocks of the caches are synchronized. Such a consistency requires the web server to include max-age or Expires information in the header, which is not commonly done. In the following, we discuss the more dominant TTL calculation (i.e. equation 3) that uses the Last-Modified time and the interaction between a web client and proxy cache.

To study the effect of the TTL calculation mismatch on the number of validation requests sent by the client, we have carried out simulations for different combinations of TTL calculations being applied to a proxy log file. Details of the log are described in the next subsection followed by the simulation results.

CONCLUSION

We conferred a client-based cache consistency theme for MANETs that depends on estimating the lay to rest update intervals of information things to line their end time. It makes use of piggybacking and pre-fetching to extend the accuracy of its estimation to scale back each traffic and question delays. We have a tendency to compare this approaches to pull-based approaches(fixed TTL and consumer polling)and to two server based approaches (SSUM and UIR). This showed that DCIM provides a much better overall performance than the opposite consumer primarily based schemes and comparable performance to SSUM. In this process traffic was reduced by using the DCIM concepts. It shows the lowest Data consistency in all the graphs, and higher traffic overhead in DCIM. For future work, investigate additional refined TTL algorithms to switch the running average formula. Extend our preliminary add to develop an entire reproduction allocation. DCIM assumes that every one node is well behaved, as problems associated with security weren't through-about .

REFERENCES

[1] Y. Huang, J. Cao, Z. Wang, B. Jin, and Y. Fens, "Achieving Flexible Cache Consistency for Pervasive Internet Access," Proc.IEEE Fifth Ann. Int'l Conf. Pervasive Computing and Comm., pp. 239-250, 2007.

[2] O. Bah at and A. Makowski, "Measuring Consistency in TTLBased Caches," Performance Evaluation, vol. 62,pp. 439-455, 2005.

[3] M. Denko and J. Titan, "Cooperative Caching with Adaptive Perfecting in Mobile Ad Hoc Networks," Proc. IEEE Int'l Conf.
Wireless and Mobile Computing, Networking and Comm. (Wino '06), pp. 38-44, June 2006.

[4] J. Jing, A. Elmagarmid, A. Hell, and R. Alonso, "Bit-Sequences:
An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," Mobile Networks and Applications, vol. 2, pp. 115-127, 1997.

[5] Q. Hum and D. Lee, "Cache Algorithms Based on Adaptive Invalidation Reports for Mobile Environments," Cluster Computing,
vol. 1, pp. 39-50, 1998.

[6] Z. Wang, S. Das, H. Chi, and M. Kumar, "A Scalable Asynchronous Cache Consistency Scheme (SACCS) for Mobile
Environments," IEEE Trans. Parallel and Distributed Systems, vol. 15, no. 11, pp. 983-995, Nov. 2004.

[7] S. Lim, W.C. Lee, G. Cao, and C. Das, "Cache Invalidation Strategies for Internet-Based Mobile Ad Hoc Networks," Computer Comm., vol. 30, pp. 1854-1869, 2007.
[8] K.S. Khorana, S. Gupta, and P. Remain, "A Scheme to Manage Cache Consistency in a Distributed Mobile Wireless Environment," IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 7, pp. 686-700, 2001.

[9] V. Cater, "Alex - A Global Filesystem," Proc. USENIX File System Workshop, pp. 1-12, May 1992.

[10] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc
Networks," IEEE Trans. Mobile Computing, vol. 5, no. 1, pp. 77-89, Jan. 2006.