Wireless Networks 8, 671–679, 2002

Predictive Caching Strategy for On-Demand Routing Protocols in Wireless Ad Hoc Networks

WENJING LOU and YUGUANG FANG *
Wireless Networks Laboratory (WINET), Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA

Abstract. Route caching strategy is important in on-demand routing protocols in wireless ad hoc networks. While high routing overhead usually has a significant performance impact in low bandwidth wireless networks, a good route caching strategy can reduce routing overheads by making use of the available route information more efficiently. In this paper, we first study the effects of two cache schemes, "link cache" and "path cache", on the performance of on-demand routing protocols through simulations based on the Dynamic Source Routing (DSR) protocol. Since the "path cache" DSR has been extensively studied, we focus in this paper on the "link cache" DSR in combination with timer-based stale link expiry mechanisms. The effects of different link lifetime values on the performance of routing protocol in terms of routing overhead, packet delivery ratio and packet latency are investigated. A caching strategy incorporating adaptive link timeout is then proposed, which aims at tracking the "optimal" link lifetime under various node mobility levels by adaptively adjusting the link lifetime based on the real link lifetime statistics. The performance of the proposed strategy is then compared with the conventional "path cache" DSR. The results show that without a timeout mechanism, a link cache scheme may suffer severe performance degradation due to the use of broken routes, while the proposed adaptive "link cache" strategy achieves significantly improved performance by reducing the routing overhead when the network traffic load is high.

Keywords: ad hoc networks, on-demand routing, dynamic source routing, timeout mechanism

1. Introduction

An ad hoc network is an infrastructureless multi-hop mobile wireless network. In an ad hoc network, there is no fixed infrastructure such as base stations that function as routers in a wireless cellular network. Each node in an ad hoc network is capable of moving independently and functioning as a router that discovers and maintains routes and forwards packets to other nodes. Due to its self-configuring, self-organizing nature, and its capability to be promptly deployed without any wired base stations or infrastructure support, ad hoc networks have been very attractive in tactical and military applications, where fixed infrastructures are not available or reliable, and fast network establishment and self-reconfiguration are required. Examples include the tactical communication in a battlefield and the disaster rescue after an earthquake. Recently, due to the availability of wireless communication devices that operate in the ISM band, the interest in ad hoc networks has extended to civilian life such as on-the-fly setup for conferencing and home-area wireless networks.

Since each node in an ad hoc network is capable of moving collectively or independently of other nodes, the network topology can be changed dramatically. Traditional Internet routing protocols are no longer effective in ad hoc networks. It presents a great challenge for a routing protocol to keep up with the frequent and unpredictable topology changes. Much work has been done since the mobile ad hoc networking (MANET) working group was formed within the Internet Engineering Task Force (IETF) to develop a routing framework for IP-based protocols in ad hoc networks. Existing ad hoc routing protocols can be generally categorized into two classes: table-driven (or proactive, such as DSDV [12] and WRP [8]) and demand-driven (or reactive, such as DSR [5] and AODV [11]) [14]. Similar to the routing protocols used in the wired network, table-driven routing protocols attempt to maintain consistent, up-to-date route information from each node to every other node, regardless of the need of such routes. They respond to topology changes by propagating updates throughout the network. An on-demand routing protocol differs in that it attempts to discover a route to a destination only when it is presented a packet for forwarding to that destination. Discovered routes are maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. Most of the performance studies indicate that on-demand routing protocols perform better than table-driven routing protocols [1,2,4]. The major advantage of the on-demand routing comes from the reduction of the routing overhead, as high routing overhead usually has a significant performance impact in low bandwidth wireless networks.

However, on-demand routing also has its disadvantages. First of all, since a route is discovered on a need basis, the packet cannot be sent before such a route has been found. Since the source node has no idea about where the destination is, the protocol has to search the entire network to find the destination. This is a very costly operation. It also adds the latency to the packet delivered. Secondly, in order to avoid the need to rediscover each routing decision for each individual packet, any on-demand routing protocol must utilize some type of routing cache to store the routes previously

* Correspondence author.

discovered. However, due to the frequent topology changes and lacking of timely update mechanisms on a regular basis, the cache itself may contain out-of-date information, implying that links in the cache are actually no longer valid. This stale data represents a liability that may degrade performance rather than improve it [6]. There is a tradeoff in using the cached information: valid data may reduce route discovery cost which will contribute to the improved performance, while stale data will cause more errors, longer delay and higher packet loss.

In this paper, we study the effects of different caching strategies on the performance of on-demand routing protocol in an ad hoc network. Two types of cache schemes, a *path cache* and a *link cache*, are investigated. A path cache is one in which each cache entry is a node list representing an entire path leading to a certain destination, while a link cache has a conventional graph data structure in which each individual link is referred to as a cached data unit. A link cache has the potential to utilize the obtained route information more efficiently. However, without a good link update mechanism, stale links may cause more route errors and consequently severe performance degradation. Marina and Das [7] studied a few techniques to improve cache correctness in DSR. Their study was based on a path cache structure. Hu and Johnson [3] conducted a comprehensive study on the caching strategies in on-demand routing protocols for wireless ad hoc networks. However, their study on the timeout mechanisms is limited to a fixed level of node mobility. A static "optimal" lifetime was assigned to each node with this mobility level. Their adaptive timeout mechanism was also limited to fine tune-up of the lifetime within the same level of node mobility and does not adapt to various node mobility levels. In this paper, we investigate two types of link update mechanisms, a static timeout mechanism and an adaptive timeout mechanism. The static timeout mechanism expires a link using a statically assigned lifetime, while in the adaptive timeout mechanism, we propose to adapt the link timeout interval to various node mobility levels based on the estimation from a moving average of real link lifetime statistics. The performance results of different cache schemes obtained from simulations will be presented.

Our simulation is based on the Dynamic Source Routing (DSR) protocol, since it is a well performed and entirely on-demand routing protocol. Previous studies have shown that a path cache DSR protocol has good performance in less "stressful" situations, i.e. smaller number of nodes and lower load and/or mobility [1,13]. In this paper, we will show that the performance of DSR under heavy traffic load situation could be much improved when incorporating a link cache scheme together with an adaptive timeout mechanism.

Although our simulations are based on DSR, the proposed adaptive link cache strategy does not tie to DSR protocol. Due to the on-demand nature, the on-demand routing protocols do not exchange routing information periodically. Even though some protocols could detect the local link broken promptly with the use of *hello* messages, the bad news (e.g., broken link) propagates to other nodes very slowly. Timer-based stale route removal strategies are also used in other routing protocols, such as AODV, in an attempt to maintain the consistence of the route cache without consuming extra network bandwidth. We believe that our results could be generalized to other protocols that use similar caching schemes.

The rest of the paper is organized as follows. Section 2 outlines the idea and methodology. We give a brief overview of the DSR protocol, discuss the different organizations of the route cache used in DSR, and then we propose the adaptive timeout mechanism for the link cache scheme. In section 3, we describe the simulation framework, and summarize the simulation results of different cache schemes in terms of routing overhead, packet delivery ratio, and end-to-end packet latency. Conclusions are drawn and future work is discussed in section 4.

2. Methodology

2.1. Overview of the DSR protocol

DSR is an on-demand routing protocol that is based on the concept of source routing. The protocol is composed of two major mechanisms, i.e. *Route Discovery* and *Route Maintenance*, and three types of route control messages, i.e. *Route Request*, *Route Reply*, and *Route Error*. When a source node in the ad hoc network attempts to send a packet to a destination but it does not already have a route to that destination in its route cache, it initiates a route discovery process by broadcasting a route request packet. This route request packet contains the source node address, the destination node address, a unique sequence number, and an empty route record. Each intermediate node, upon receiving a route request for the first time, will check in its own route cache. If it has no route to the destination, the intermediate node will add its own address to the route record and rebroadcast the route request. If it has a route to the destination in its route cache, the intermediate node will append the cached route to the route record and initiates a route reply back to the source node. The route reply contains the complete route record from the source to the destination. The intermediate node ignores the latecomers of the same route request by examining the sequence number. If the node receiving the route request is the destination node, it will copy the route record contained in the route request and send a route reply back to the source. In most simulation implementations, the destination node will reply to all the route requests received as DSR is capable of caching multiple paths to a certain destination and the replies from the destination most accurately reflect the up-to-date network topology.

Due to the node movement, the routes discovered may no longer be valid over time. The route maintenance mechanism is accomplished by sending route error packets. When a link is found broken, a route error packet is sent back from the node that detects the link failure back to the source node. Each node, upon receiving the route error message, removes from its cache all the routes that contain the broken link [5]. Some ad hoc routing protocols, such as WRP [8]

and AODV [11], use the periodic local broadcasts of the *hello* messages to ensure the local connectivity. Nodes learn of the existence of a neighbor from receiving or overhearing the packets transmitted by that node. When a node is not sending anything, it must send a *hello* message within a specified time period. Otherwise, failure to hear from a neighbor indicates a broken link between the two. DSR does not incorporate such local connectivity maintenance mechanism. In DSR, each node transmitting the packet is responsible for confirming that the packet has been received by the next hop along the source route. This can be done by either a link layer acknowledgement (as in IEEE 802.11), or a "passive acknowledgement" (in which the first transmitting node confirms the receipt at the second node by overhearing the second node transmitting the packet to the third node), or a DSR-specific software acknowledgement returned by the next hop. Thus, once a route enters the cache, the failure of the route can only be detected when it is actually used to transmit a packet but fails to confirm the receipt by the next hop.

Besides the aforementioned basic functions, more optimization mechanisms are proposed and added to DSR protocol. These optimizations include gratuitous route replies, salvaging, gratuitous route errors, snooping, tapping, etc. [6]. Most of the optimizations are included in our simulation implementation.

2.2. Cache organizations

In DSR, the route returned to the source is a complete path leading to the destination. By caching each of these paths separately, a "path cache" organization can be formed. A path cache is very simple to implement. When a route is needed, the path cache data structure can be efficiently searched for any path leading to that destination. This type of cache organization has been extensively studied through simulations [1,7,13]. In this paper, we consider an alternative type of organization, a "link cache", in which each node keeps a graph data structure representing this node's current view of the network topology, the route returned to this node is decomposed into individual links and represented in the graph data structure [3]. When a route is needed, a graph search algorithm, such as the Dijkstra shortest path algorithm or the breath-first-search (BFS) shortest path algorithm, is executed to find a path to the destination.

Compared with a path cache scheme, a link cache has the potential to utilize the route information more efficiently. Given the same amount of route reply information, the routes existing in a path cache can always be found in a link cache, while by connecting individual links differently the link cache may have new or better routes that a path cache does not include. In a path cache, a complete route will be removed (or is truncated before the broken link, depending on the path representation in the cache) when one of its links breaks. While in a link cache, only the broken link is removed. The rest of the links on that route are still available to form new routes. Consider the situation illustrated in figure 1, the link cache will produce a new and better path to E as A → C → E, which

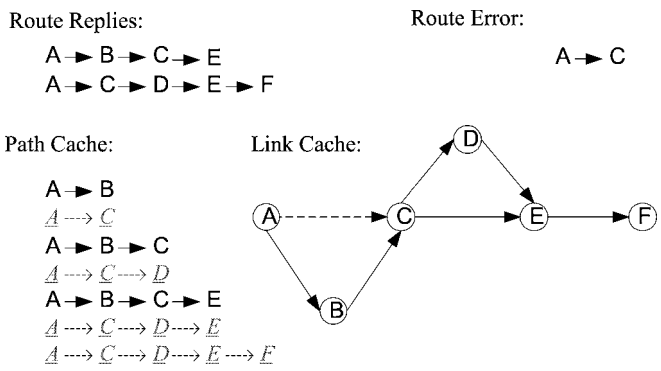


Figure 1. Illustration of path cache and link cache.

does not exist in the path cache. When link A → C is broken, there will be no route to destination D and F exist in a path cache. A route discovery process has to be initiated if a route to destination D or F is desired. While by removing only link A → C, a link cache still has route A → B → C → D or A → B → C → E → F to the destination D and F. A route discovery can be avoided. Therefore, with a link cache, potential reduction in the costly route discovery operation can be expected.

2.3. Link timeout mechanisms

As nodes in an ad hoc network are capable of moving independently, a link has a limited lifetime. Current existing link is no longer valid when the two end nodes moving out of the transmission range of each other. Due to the on-demand manner of the routing protocol, the link status will not be updated until they are used. However, using an actually broken route will cause a number of route errors to be generated and potential packet loss. So taking advantage of using active links individually has to be combined with a mechanism that removes stale links timely to avoid route errors. A natural choice is to combine a timeout policy to the link cache such that each link is given an appropriate lifetime when it enters the link cache, and is removed when its lifetime is running out. The lifetime estimation becomes a critical issue for such a link cache scheme. The lifetime assigned to the link should properly reflect the expected value of its real lifetime. If this value is assigned too small, links expire too soon before they really break, more costly route discoveries would have to be performed. On the other hand, if the value is too large, links break early before the timers expire, more route errors will be caused, which would also degrade the overall performance of the protocol.

In this paper, we first study the static lifetime assignment, in which when a link enters the link cache, it is assigned a predefined static lifetime T_1 seconds. During its lifetime, if the link is used to send packets, the lifetime of this link will be adjusted so that it will not expire in the future T_2 seconds. Then, based on the observations from the static lifetime experiments, we propose and study an adaptive lifetime estimation scheme that adaptively adjusts the link lifetime based on the moving average of the pervious collected lifetime statistics.

In our adaptive link lifetime scheme, each link (i, j) in the link cache is associated with three clock type attributes, *born*, *lastUsed*, and *liveTo*. Attribute *born* indicates the time when a new link enters the link cache. It is updated when a route reply is received and a new link is found in the route. Attribute *lastUsed* is the time stamp when the link is last used to forward a packet. Attribute *liveTo* is the predicted time at or after which the link expires. The statistical lifetime data are collected whenever a link is removed from the link cache. There are two situations that will cause a link to be removed from the link cache – a route error is received indicating that link is broken, or the *liveTo* attribute associated with that link expires. If a link is removed because of the reception of a route error, the lifetime l is calculated as

$$l = \mathit{CurrentTime}() - \mathit{link}[i, j].\mathit{born},$$

or if it is removed because of timeout, lifetime l is calculated only if this link has ever been used during its lifetime,

$$l = \mathit{link}[i, j].\mathit{lastUsed} - \mathit{link}[i, j].\mathit{born}.$$

LIFETIME is the variable indicating the estimation of the link lifetime. It is initially assigned a static value and is adjusted dynamically using a moving average method whenever a lifetime datum l is collected,

$$\mathit{LIFETIME} = (1 - \alpha) \cdot \mathit{LIFETIME} + \alpha \cdot l,$$

where α is the first order moving average parameter, which indicates the weight of the new datum in the total average. In our simulation, we set the initial *LIFETIME* as 50 seconds and α as 0.01.

The effects of different static lifetime values on the performance of routing protocol are investigated by simulations, which are reported in the next section.

3. Simulation and results

3.1. Simulation framework

The simulation of our link cache DSR protocol is implemented within the GloMoSim library [15,16]. The GloMoSim library is a scalable simulation environment for wireless network systems using the parallel discrete-event simulation language called PARSEC. The link layer model is the Distributed Coordination Function (DCF) of the IEEE 802.11 wireless LAN standard. The radio model uses the channel characteristics similar to Lucent's WaveLAN product. Radio propagation range for each node is 250 m and channel capacity is 2 Mbits/s. The network simulated consists of 50 nodes. The simulation area is 700×700 m^2. Each simulation is executed for 15 simulated minutes.

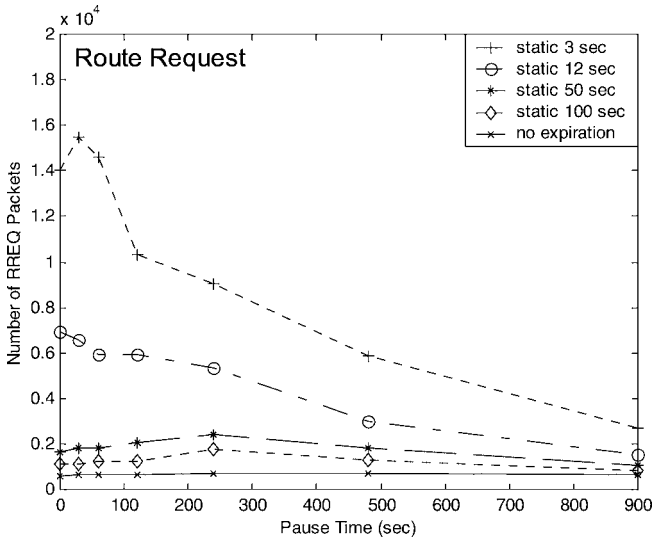
The random waypoint mobility model is used in the simulation [1]. In this model, a node selects a destination randomly within the simulated territory, moves to that destination at a speed uniformly distributed in $[v_{\min}, v_{\max}]$ m/s, stops there for a predefined *pause time* and then repeats this behavior for the duration of the simulation. In our simulations, $[v_{\min}, v_{\max}]$ is fixed to $[0, 20]$. The different node mobility levels are achieved by changing the values of pause time.

The traffic used in this simulation is CBR UDP sessions. The source–destination pairs are chosen randomly among the 50 nodes. The number of communication sessions is varied to change the offered load in the network. All the data packets are 64 bytes and are sent at a speed of 4 packets/s. The reason that we choose 64 bytes small packet size is because our focus is on routing protocol's capability of tracking the topology change. Small packet size will factor out the effects of other reasons such as the network congestion [1].

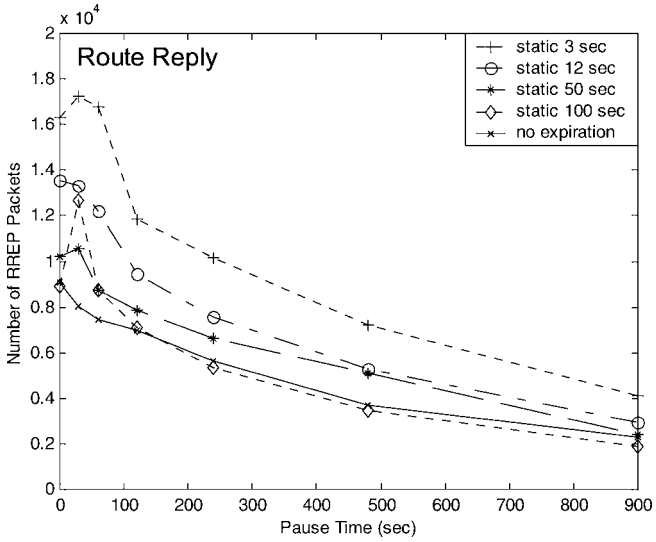
We evaluate the performance of the link cache DSR protocol with static lifetime assignments and the proposed adaptive lifetime scheme. With static lifetime assignments, we execute multiple simulations for different (T_1, T_2) values as (3, 1), (6, 1), (12, 2), (25, 3), (50, 5), (100, 10), (900, –). For each (T_1, T_2) value, we execute multiple simulation runs with various traffic load conditions (10, 20, 30, 40, 50 sources) and various node mobility levels (pause time = 0, 30, 60, 120, 240, 480, 900 s). For comparison purpose, we also evaluate the conventional path cache DSR protocol. The traffic load and the node mobility scenarios are identical across different variations of DSR protocol.

3.2. Effects on routing overhead

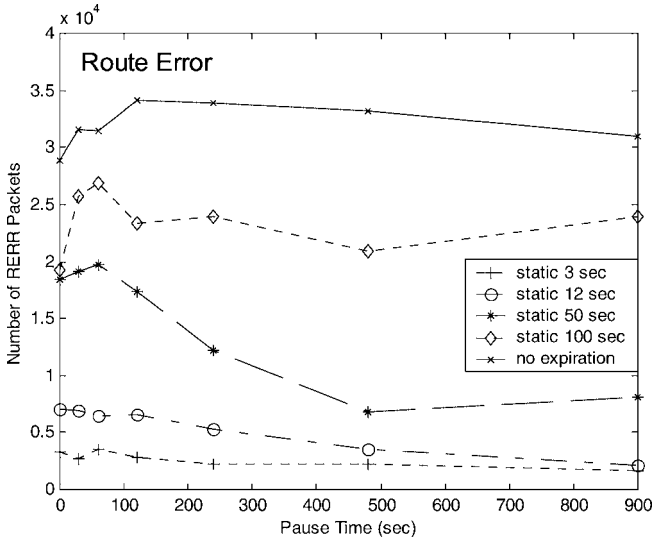
We first examine the effects of different cache schemes on the routing overhead generated. The routing overhead is calculated as the number of control packets transmitted by the protocol. It is counted as per hop basis. There are three types of routing packets, e.g., route requests, route replies, and route errors. Figure 2 plots the amount of each type of control packets generated when using different static lifetime values. The results confirm our expectation that small values of lifetime cause increased number of route requests but decreased number of route errors, while large values of lifetime cause decreased number of route requests but increased number of route errors. When the lifetime is within an appropriate range (in our experiments, $6 \leqslant T_1 \leqslant 50$), the overall control messages are quite balanced. However, when the lifetime value becomes very large (in our experiments, $T_1 \geqslant 50$), the dramatically increased route errors would overwhelm the slightly decreased route requests. The total number of control messages increases significantly. Figure 3 gives the comparison of the overall routing overhead generated by each variation of the protocol. We observe that, in general, with an appropriate static link lifetime, the number of control packets used by the link cache scheme is less than that used by the path cache scheme. With the increased network traffic load, the reduction becomes more obvious. However, without an appropriate timeout mechanism, the routing overheads are quite high. Among them, the majority is route error messages caused by the use of stale routes. We also observe that the proposed adaptive cache scheme tracks the "optimal" link lifetime quite well. Although it is not always optimal, it keeps the routing overhead "sub-optimally" low under various mobility level conditions.



(a)



(b)



(c)

Figure 2. Separated routing overhead for a 50 sources scenario: (a) Route Request, (b) Route Reply and (c) Route Error.

The above observations indicate that a link cache scheme, particularly the link cache scheme with adaptive lifetime estimation, could make use of the available route information more efficiently. When the network traffic load is high, more route replies and more promiscuously overheard route information can be collected. The link cache can maintain more accurate and more up-to-date information of which the link cache scheme can take advantage to reduce the route discovery processes. The reduced routing overhead, especially in heavy traffic situation, contributes to the improved performance, as discussed in the following subsections.

3.3. Effects on packet delivery ratio

In this subsection, we evaluate the application level performance metric – packet delivery ratio (the end-to-end throughput). The packet delivery ratio is the fraction of packets that are received at corresponding destination over those sent at the source. It is the most important metric to evaluate the performance of an ad hoc routing protocol. There are two major situations that a packet may drop. One is due to the stale routes. A stale route in the cache may direct a packet to an actually broken link. When a node fails to forward a packet over a broken link, it will try to savage the packet by looking up in its own route cache for an alternative route to the packet's destination. If the alternative path does not exist or the packet has already been salvaged before, the packet is dropped. To reduce such packet drops, small lifetime value is preferred because small lifetime value could expire stale routes and reduce the chances that a stale route is used. The other type of packet loss is due to the heavy collisions in the MAC layer that cause a packet drop after failing a certain number of attempts to transmit the packet. A route discovery process can cause a large number of route requests and route replies generated within a short time, thus cause increased interference to data traffic at MAC layer. When the traffic load is high, the packet loss caused by collisions becomes more severe. To reduce the packet loss due to this reason, a large lifetime value is preferred because a large lifetime value could minimize the number of route discovery performed. Figure 4 summarizes the performance of packet delivery ratio under various network traffic loads and node mobility levels. We observe that, in general, the link cache scheme outperforms the path cache scheme when the network load is high. The heavier the network load, the clearer the trend, and the wider the performance gap. However, when the traffic load is low, the performance comparison between the two types is not certain. A link cache does not always show advantage over the path cache, as shown in 10 sources case.

If we use the packet delivery ratio as the performance index, we also observe that the "optimal" link lifetime values vary at different node mobility levels. Nodes with high mobility tend to perform better with shorter lifetime while nodes with low mobility tend to perform better with a little bit longer lifetime. No single static lifetime can provide optimal perfor-

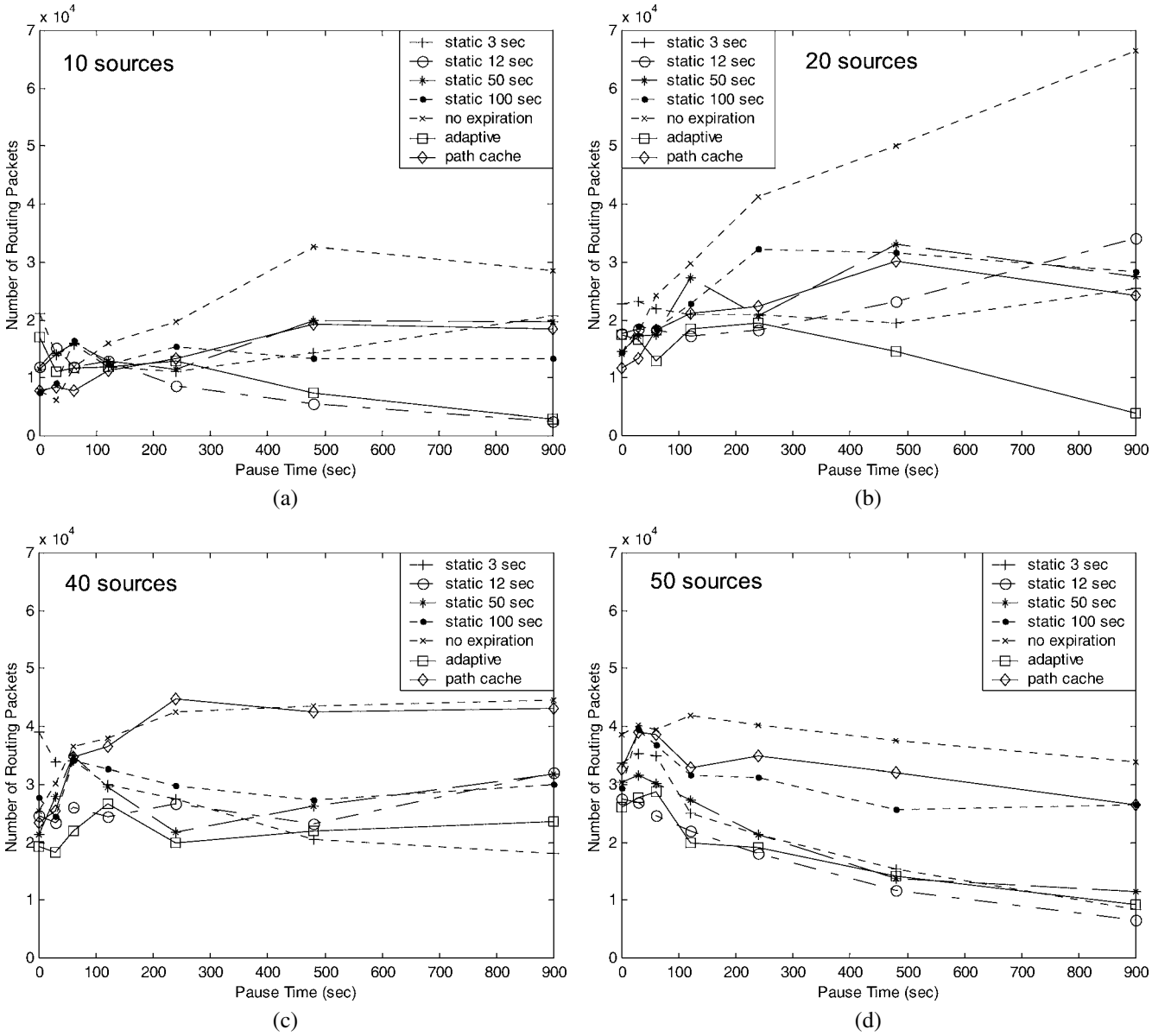


Figure 3. Comparison of total routing overhead for scenarios of (a) 10, (b) 20, (c) 40 and (d) 50 sources.

mance under all circumstances. Again we observe that the adaptive cache scheme could keep near optimal performance under various mobility scenarios. By studying the performance of packet delivery ratio and the routing overhead together, we find that the performance of packet delivery ratio in high traffic load scenarios is significantly affected by the routing overhead in ad hoc networks.

3.4. Effects on packet latency

Packet latency is another important performance metric. The packet latency is also called end-to-end delay, which is the latency between a packet sent at the source and received at the destination. The packet latency is only calculated for packets that are successfully delivered. Besides the ordinary transmission delay, propagation delay, and queuing delay, which widely exist in all IP networks, there are two types of latency caused particularly by ad hoc on-demand routing protocols. One is the latency the protocol takes to discover a route to a destination when there is no known route to that destination. This type of latency is due to the on-demand behavior of the routing protocol and exists in all such protocols. The other one is the latency for a sender to "recover" when a route used breaks. The latency resulting from broken routes could be very large because the amount of latency is the addition of the following three parts, the time for a packet travel along the route to the node immediately before the broken link, the time for that node to detect the broken link, and the time for a route error message to travel from that node back to the source node. Among them, the time to detect a broken link could be very large because the failure of the node can only be determined after having made a certain number of attempts to

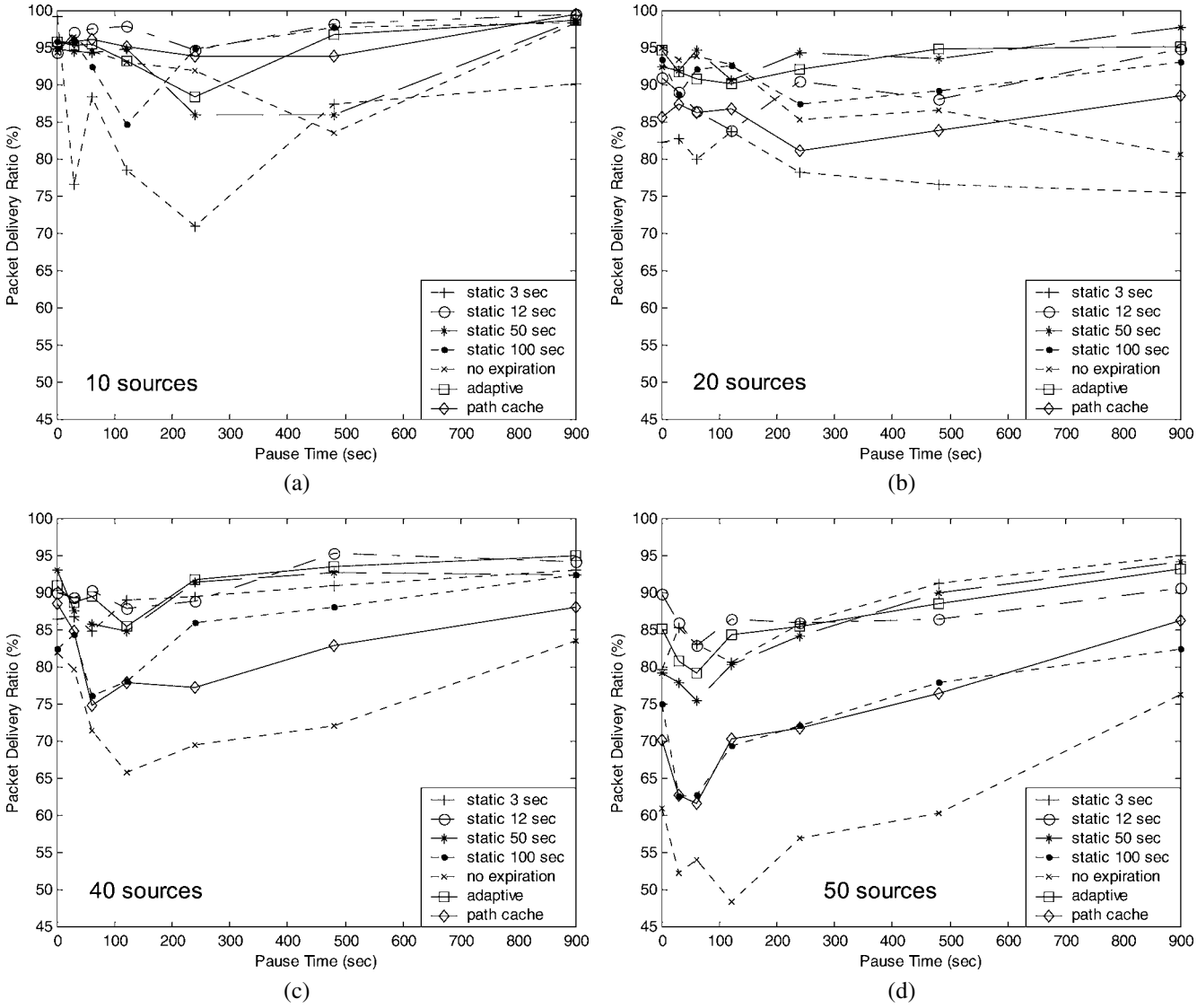


Figure 4. Comparison of packet delivery ratio for scenarios of (a) 10, (b) 20, (c) 40 and (d) 50 sources.

transmit the packet over the broken link but failed to receive a passive or explicit acknowledgement of success. Figure 5 shows the performance comparison of packet latency across different variations of DSR protocol. We observe that, with an appropriate link lifetime, the packet latency incurred by the link cache scheme is kept relatively low compared to the path cache scheme. Especially when the network traffic load grows high, the advantage of the link cache becomes clearer. Since the small link lifetime value tends to expire links earlier than they actually break, the possibility of using broken links are reduced, thus the latency caused by broken routes is minimized. Therefore, we observe that small lifetime values result in low packet latency. Correspondingly, without an appropriate timeout mechanism (e.g., the link lifetime assigned is too large), packets suffer abnormally large latency because too many broken routes are used. Again we observe that the proposed adaptive cache scheme maintains comparably low latency under various mobility level conditions.

4. Conclusions and future work

In this paper, we study the effects of the route cache schemes on the performance of on-demand routing protocols in ad hoc networks. We base our simulation on DSR, the well-evaluated on-demand routing protocol in ad hoc networks. We first study the "link cache" performance with static lifetime assignments. The results indicate that an appropriate timeout mechanism is critical on the performance of such a link cache scheme. A link cache with an appropriate timeout mechanism could make use of the available route information more efficiently, thus improve the protocol performance. However, without an appropriate timeout mechanism, a link cache may cause dramatically increased route errors and consequently severe performance degradation. We also find that in general the link cache scheme performs better when network traffic load is high. Based on these observations, we propose an adaptive link timeout mechanism in combination with a "link cache" organization (the adaptive link cache scheme)

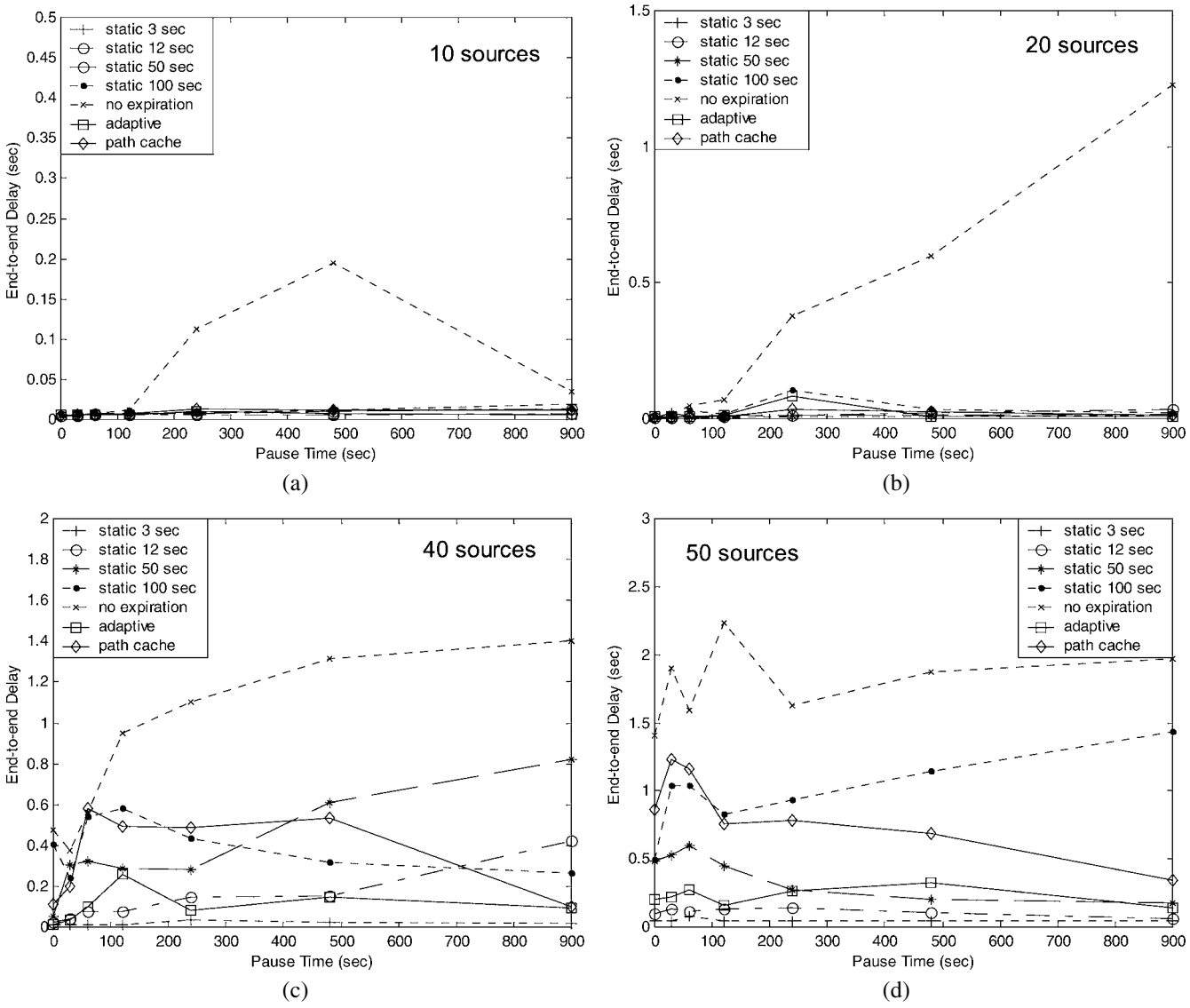


Figure 5. Comparison of end-to-end delay for scenarios of (a) 10, (b) 20, (c) 40 and (d) 50 sources.

for DSR. The adaptive link lifetime estimation scheme aims at tracking the "optimal" link lifetime under various node mobility conditions. The performance of the proposed cache strategy is compared with the conventional "path cache" DSR. The results show that when the number of traffic sources increases, the proposed "link cache" DSR outperforms the "path cache" DSR, with wider performance gap with increasing load.

Previous studies have shown that a path cache DSR protocol has good performance in less "stressful" situations, i.e. smaller number of nodes and lower load and/or mobility. In this paper, all our simulation results show that with a link cache, the performance of DSR under more stressful situation can be much improved. As the cache organization is a local implementation decision at each node, all the protocol control messages and route discovery and maintenance mechanisms remain the same, we suggest that by switching between the two types of cache organizations dynamically in response to the network load condition might provide a good way to improve the overall performance of the DSR protocol. This issue will be investigated in the future.

Acknowledgement

This work was supported in part by the Office of Naval Research Young Investigator Award under contract N000140210464 and National Science Foundation Faculty Early Career Development Award under the contract ANIR0093241.

References

[1] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocol, in: *The 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (October 1998) pp. 85–97.

[2] S.R. Das et al., Comparative performance evaluation of routing protocols for mobile ad hoc networks, in: *The 7th International Conference*

on Computer Communication and Networks (IC3N) (October 1998) pp. 153–161.

[3] Y.-C. Hu and D.B. Johnson, Caching strategies in on-demand routing protocols for wireless ad hoc networks, in: *The 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (August 2000).

[4] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek and M. Degermark, Scenario-based performance analysis of routing protocols for mobile ad hoc networks, in: *The 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (August 1999) pp. 195–206.

[5] D.B. Johnson, D.A. Maltz, Y.-C. Hu and J.G. Jetcheva, The dynamic source routing protocol for mobile ad hoc networks, IETF Internet draft (November 2001) `draft-ietf-manet-dsr-06.txt`

[6] D.A. Maltz, J. Broch, J. Jetcheva and D.B. Johnson, The effects of on-demand behavior in routing protocols for multihop wireless ad hoc networks, IEEE Journal on Selected Areas in Communications 17(8) (August 1999) 1439–1453.

[7] M.K. Marina and S.R. Das, Performance of routing caching strategies in dynamic source routing, in: *2001 International Conference on Distributed Computing Systems Workshop* (2001).

[8] S. Murphy and J.J. Garcia-Luna-Aceves, An efficient routing protocol for wireless networks, Mobile Networks and Applications, Special Issue on Routing in Mobile Communication Networks (October 1996) 183–197.

[9] M.R. Pearlman, Z.J. Hass, P. Sholander and S.S Tabrizi, On the impact of alternate path routing for load balancing in mobile ad hoc networks, in: *Proceedings of the 8th International Conference on Computer Communication and Networks (IC3N)*, Boston (October 1999).

[10] G. Pei, M. Gerla and T. Chen, Fisheye state routing: A routing scheme for ad hoc wireless networks, in: *International Conference on Communications (ICC)* (June 2000).

[11] C.E. Perkins, E.M. Belding-Royer and S.R. Das, Ad hoc on-demand distance vector (AODV) routing, IETF Internet draft (November 2001) `draft-ietf-manet-aodv-09.txt`

[12] C.E. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, Computer Communication Review (October 1994) 234–244.

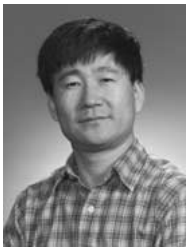
[13] C.E. Perkins, E.M. Royer, S.R. Das and M.K. Marina, Performance comparison of two on-demand routing protocols for ad hoc networks, IEEE Personal Communications (February 2001) 16–28.

[14] E.M. Royer and C.-K. Toh, A review of current routing protocols for ad hoc mobile wireless networks, IEEE Personal Communications (April 1999) 46–55.

[15] M. Takai, L. Bajaj, R. Ahuja, R. Bagrodia and M. Gerla, GloMoSim: a scalable network simulation environment, Technical report 990027, UCLA, Computer Science Department (1999).

[16] M. Takai, J. Mertin and R. Bagrodia, Effects of wireless physical layer modeling in mobile ad hoc networks, in: *ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)* (October 2001).

Wenjing Lou received the B.E. degree and M.E. degree in computer science and engineering from Xi'an Jiaotong University, China, in 1993 and 1996, respectively. She received the M.A.Sc. degree in computer communications from Nanyang Technological University, Singapore, in 1998. From December 1997 to July 1999, she worked as a Research Engineer in the Network Technology Research Center, Nanyang Technological University. She is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of Florida. Her research interests include wireless ad hoc networks, routing protocols in both wired and wireless networks, and network security. Wenjing Lou is a member of Tau Beta Pi and a student member of the IEEE.
E-mail: wjlou@ufl.edu



Yuguang Fang received a Ph.D. degree in systems and control engineering from Case Western Reserve University in January 1994, and a Ph.D. degree in electrical engineering from Boston University in May 1997. From June 1997 to July 1998, he was a Visiting Assistant Professor in the Department of Electrical Engineering at the University of Texas at Dallas. From July 1998 to May 2000, he was an Assistant Professor in the Department of Electrical and Computer Engineering at the New Jersey Institute of Technology. Since May 2000, he has been an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Florida. His research interests span many areas including wireless networks, mobile computing, mobile communications, automatic control, and neural networks. He has published over sixty papers in refereed professional journals and conferences. He has received the National Science Foundation Faculty Early Career Development Award in 2001 and the Office of Naval Research Young Investigator Award in 2002. He is listed in Marquis Who's Who in Science and Engineering, Who's Who in America and Who's Who in the World. Dr. Fang has engaged in many professional activities. He is a senior member of the IEEE and a member of the ACM. He is an Editor for IEEE Transactions on Communications, an Editor for IEEE Transactions on Wireless Communications, an Editor for ACM Wireless Networks, an Area Editor for ACM Mobile Computing and Communications Review, an Associate Editor for Wiley International Journal on Wireless Communications and Mobile Computing, and Feature Editor for Scanning the Literature in IEEE Personal Communications. He has also been actively involved with many professional conferences such as ACM MobiCom'02 (Co-Chair for Student Travel Award Committee), ACM MobiCom'01, IEEE INFOCOM'03, INFOCOM'00, INFOCOM'98, IEEE WCNC'02, WCNC'00 (Technical Program Vice-Chair), WCNC'99, and International Conference on Computer Communications and Networking (IC3N'98) (Technical Program Vice-Chair).
E-mail: fang@ece.ufl.edu