Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad hoc Networks

Alvin Valera*, Winston K.G. Seah*† and SV Rao†
*Department of Computer Science
School of Computing, National University of Singapore, Singapore 117543
Email: alvincva@comp.nus.edu.sg
†Institute for Communications Research
20 Science Park Rd, #02-34/37 Teletech Park, Singapore 117674
E-mail: {winston,raosv}@icr.a-star.edu.sg

***Abstract*--- A mobile ad hoc network is an autonomous system of infrastructureless, multihop wireless mobile nodes. Reactive routing protocols perform well in such an environment due to their ability to cope quickly against topological changes. In this paper, we propose a new routing protocol called Caching and Multipath (CHAMP) Routing Protocol. CHAMP uses *cooperative packet caching* and *shortest multipath routing* to reduce packet loss due to frequent route breakdowns. Simulation results reveal that by using a five-packet data cache, CHAMP exhibits excellent improvement in packet delivery, outperforming AODV and DSR by at most 30% in stressful scenarios. Furthermore, end-to-end delay is significantly reduced while routing overhead is lower at high mobility rates.**

I. Introduction

A *mobile ad hoc network* is an autonomous system of multihop, wireless mobile nodes that does not require basestations or any fixed infrastructure. It is characterized by dynamic topologies, bandwidth-constrained, variable capacity links, energy constrained operation and limited physical security [1]. The lack of infrastructure, in combination with multihop connections and constantly changing topology pose difficult challenges on the routing protocol; foremost among them is how to deliver data packets while incurring the least routing overhead possible.

Over the last few years, many routing protocols for mobile ad hoc networks have been proposed [2], [3], [4], [5], [6], [7], [8]. These protocols can be broadly classified into three categories, namely, *proactive, reactive,* and *hybrid*. Studies show that reactive routing protocols perform better in terms of packet delivery and routing overhead especially in the presence of mobility due to the ability of these protocols to quickly detect link failures [9], [10], [11].

In reactive routing, a node declares a link to a neighbor as down if it fails to forward a packet to this neighbor. This may cause one or more packets (at the node that encountered the forwarding failure) to become undeliverable. To avoid dropping such packets, DSR proposed an optimization known as ''packet salvaging'' [6]. In this optimization, the node that encountered the forwarding failure searches its route cache for alternative routes. If a route is found, undeliverable packets are forwarded through the alternative path. However, when no alternative path is found, these undeliverable packets are simply discarded. It is evident that in situations where link failures are frequent, this simple mechanism may result in increased packet loss as there is no guarantee that packet salvaging will always be successful.

In this paper, we introduce *cooperative packet caching*, a technique that exploits temporal locality in dropped packets, aimed at reducing packet loss due to route breakage. Every node maintains a small buffer for caching data packets that pass through it. When a downstream node encounters a forwarding error, an upstream node with the pertinent data in its buffer and alternative route can retransmit the data. For this strategy to be effective, nodes must store multiple routes to every active destination. Hence, we propose a simple route discovery mechanism that selects the *shortest multipath routes*.

The rest of the paper is organized as follows: Section II discusses the property of temporal locality in dropped packets. This property forms the basis of cooperative packet caching. Section III presents CHAMP, a routing protocol that implements cooperative packet caching and shortest multipath routing. In Section IV, we discuss the simulation models based on *ns-2*. In Sections V and VI, we evaluate the performance of CHAMP and compare it with AODV and DSR. In Section VII, we present a survey of related work. In Section VIII, we finally state our conclusions.

II. Exploiting Locality in Ad hoc Networks

The use of caching to improve performance is not a new concept. In fact, caching has been around since 1965 when Wilkes [12] introduced it to bridge the speed gap between processor and main memory. Basically, a cache is a small but fast memory that stores data for use in the near future. It exploits the property of locality in memory references in order to reduce latency and increase memory bandwidth.

Two kinds of locality have been observed - *temporal* and *spatial* [13]. Spatial locality is the property whereby an access to a memory location indicates that a nearby location will very likely to be accessed in the near future. Temporal locality is the property whereby an access to a memory location indicates that the same location will very likely to be accessed again soon. It must be emphasized that without these properties,

0-7803-7753-2/03/$17.00 (C) 2003 IEEE

(that is, if memory accesses are totally random and independent) caching will not improve performance since most cache accesses will lead to ''misses'' which results in the processor always accessing the main memory.

At present, caching is used not only in single-processor environments but also in distributed systems [14]. The underlying principle, though, is still locality. In mobile ad hoc networks, Castaneda and Das [15] first investigated spatial locality in the context of node mobility. They observed that when a mobile node moves, ''it cannot move too far too soon''. Query localization techniques exploit this property to lower routing overhead during route discovery and repair. Instead of network-wide flooding, route requests are flooded only to a limited region that is previously part of a valid route [15]. The reasoning is that the destination (or some nodes with valid routes) can never be too far too soon from these nodes, hence there is a big probability of finding it.

We now consider this situation: Suppose that some node i is along some route from source h to destination j. Suppose further that whenever a node fails to forward a data packet to its next hop, it drops the packet and sends a route error message to the source using the reverse route. Observe that whenever i receives a route error from a downstream node k, the error indicates the dropping of a data packet that source h has *recently* sent and i has *recently* forwarded. Clearly, dropped packets exhibit the property of temporal locality, that is, ''a dropped packet is a recently sent packet''. Therefore, if i has a buffer for caching forwarded data packets, even if the buffer is small, there is a big probability that the packet is still in that buffer. And if i has some other route to j, it can ''salvage'' the dropped packet and need not forward the route error upstream.

Although additional storage overhead is required for the data cache and multiple routes, this technique, which we call *cooperative packet caching*, can reduce packet loss due to route breakdowns. In existing reactive routing protocols, only the node encountering the error can salvage or retransmit a data packet. Cooperative packet caching enables more nodes to salvage a dropped packet, or in essence, *packet salvaging is distributed*.

For cooperative packet caching to be effective, every node must maintain at least two routes to every active destination. One major problem of multiple path routing is the large routing overhead generated during route search and maintenance. A study revealed that TORA [3], a multiple path protocol, generates more than 50 times the routing overhead of AODV and DSR [9]. One solution to eliminate maintenance overhead is to use data packets in place of the ''keep-alive'' packets by spreading the packets over all the routes. However, per-packet spreading creates another problem. The number of packets that arrive out-of-order in the destination will increase, possibly degrading the performance of certain applications. To address this problem, we only select shortest multipath routes [16] with equal lengths.

III. CHAMP Routing Protocol

We briefly describe the routing protocol in this Section and illustrate the important features, particularly shortest multipath route discovery and cooperative packet caching. A complete discussion of the routing algorithm including the proof of correctness is presented in a previous work [17].

An ad hoc network is represented as a graph $\mathbf{G} = (\mathbf{N}, \mathbf{L})$, where $\mathbf{N}$ is the set of nodes and $\mathbf{L}$ is the set of edges or links. Any node $i \in \mathbf{N}$ can act both as a router or data source. Let $\mathbf{N}^i$ be the set of neighbors of i defined as the set nodes where i has direct bidirectional connectivity.

The successor set at node i for each active destination j, denoted by $\mathbf{S}_j^i \subseteq \mathbf{N}^i$, is defined as the set of nodes that can be used by i as a next hop for packets destined to j. The length of any route from $k \in \mathbf{S}_j^i$ to j is D_j^k. A unique property of CHAMP is that it only includes a node k in $\mathbf{S}_j^i$ if the length of the route from k to j is equal to the shortest path.

A. Data Structures

Every node maintains two data structures: a *route cache* to contain forwarding information; and a *route request cache* for storing recently received and processed route requests. The route cache at node i is a list containing an entry for each active destination j. Each entry contains the following: destination identifier (j), distance to the destination (D_j^i), set of successor or next hop nodes to the destination ($\mathbf{S}_j^i$), the time each successor node k was last used for forwarding ($ltu_{jk}^i, \forall k \in \mathbf{S}_j^i$), and number of times each successor node k is used ($use_{jk}^i, \forall k \in \mathbf{S}_j^i$). A route entry which has not been used for more that $RouteLifeTime$ seconds is deleted.

The route request cache at node i is a list containing an entry for every unique route request received and processed. Each entry contains the following: source identifier of the route request (h), identifier of the node being searched (j), sequence number of the request (sn), minimum forward count ($minfc_{ji}^h(sn)$), set of nodes that forwarded the same request with $fc = minfc_{ji}^h(sn)$ ($\mathbf{P}_{ji}^h(sn)$), and status of the route request ($status_{ji}^h(sn)$) which can either be $Replied$ or $NotReplied$.

In addition to the two data structures, every node also maintains two first-in first-out buffers: a *send buffer* for storing packets waiting for routes; and a *data cache* for storing recently forwarded data packets.

B. Route Discovery

CHAMP operates on-demand, that is, node i maintains $\mathbf{S}_j^i$ only if there data packets to be sent to j. Its route discovery is similar to the ''diffusing computations'': given a direct acyclic graph (DAG), each node computes its distance based on the distance reported by the downstream nodes and reports its distance to its upstream nodes [18].

A source node h initiates route discovery when it has data to send to j but it has no available route. Node h floods the network with a RREQ (route request message) for j. This establishes a DAG rooted at h. When j receives a RREQ, it

sends back a RREP (route reply message) to i through some nodes that is a subset of the DAG rooted at h.

Every RREQ from h to j has a forward count fc field, which is initialized to zero by the source and incremented by one every time the message is retransmitted. The first time i receives a RREQ from h to j, it initializes $minfc^h_{ji}$ to fc and $\mathbf{P}^h_{ji}$ to the previous hop of the message. Every time i receives a request with $fc = minfc^h_{ji}$ (meaning the request traversed a path of the same length from h to this node), it includes the previous hop of the message in $\mathbf{P}^h_{ji}$. If i receives a request with $fc < minfc^h_{ji}$ (meaning the request traversed a shorter path from h to this node), it resets $minfc^h_{ji}$ to fc and $\mathbf{P}^h_{ji}$ to the previous hop of the message. The set $\mathbf{P}^h_{ji}$ contains the identifiers of nodes that can receive a corresponding RREP from i, if i sends one.

When a destination node j receives a RREQ, it immediately sends back a RREP if $fc \leq minfc^h_{jj}$. Every RREP explicitly specifies the set of nodes $\mathbf{P}$ that can accept it. The destination node j initializes this field to the previous hop of the RREQ, effectively indicating that the RREP is only intended for this node. Every RREP also has a hop count field hc initialized to zero by the destination.

A node i processes a RREP if $i \in \mathbf{P}$. Node i then accepts the route in RREP if $hc \leq D^i_j$ or its existing routes to j have not been used for more than $RouteFreshTime$ seconds and provided that the number of routes to j is less than or equal to $MaxRoutes$. If the received route is shorter ($hc < D^i_j$) or existing routes to j have not been used for more than $RouteFreshTime$ seconds, $\mathbf{S}^i_j$ is reset to contain the previous hop of RREP. Node i then computes its distance $D^i_j \leftarrow hc$ and forwards the message to its upstream nodes by setting $\mathbf{P} \leftarrow \mathbf{P}^h_{ji}$ and incrementing hc by one if the corresponding request has not been replied yet. This process is repeated until the RREP reaches the source h.

C. Data Forwarding

In CHAMP, data packets are identified by the source identifier and a source-affixed sequence number. They also include their respective previous hop in their header, serving as a ''pointer'' to the upstream node which cached the same data.

When forwarding a data packet, a node i chooses the least used next hop neighbor k. This spreads the packets over all the routes in a round-robin fashion. Node i then saves a copy of the packet in its data cache, sets the previous hop field to its address, and then forwards the data packet to the chosen next hop. If i has no route to j and it is the source of the packet, it saves the packet in its send buffer and performs a route discovery. However, if i is not the source, it simply drops the packet and broadcasts a RERR containing the header information (source, destination, previous hop and sequence number) of the dropped packet.

D. Route Maintenance

Nodes rely on data packet acknowledgment provided by the link layer to determine the state of a link. Since packets are forwarded in a round-robin fashion, all links are periodically refreshed. Route maintenance occurs only when a node i loses all its active routes to some destination j after a data forwarding failure.

A link (i, k) is declared as ''down'' when node i does not receive an acknowledgment from the next hop node k after forwarding a data packet to k. When this occurs, k is deleted from $\mathbf{S}^i_j$. If i has another route to j, it forwards all undeliverable packets through this route. If i has no other route to j, i broadcasts a RERR containing the header information of all data packets (excluding packets that are originating from this node) that cannot be delivered as a result of the link failure. If there are undeliverable packets originating from this node, i performs a route discovery.

E. Packet Salvaging from Data Cache

Recall that it is possible for the RERR to contain header information of one or more data packets. Before parsing the RERR received from k, i creates a new RERR message that it will propagate upstream should it fail to salvage any packet. For every packet referred to in the message, node i performs the following:

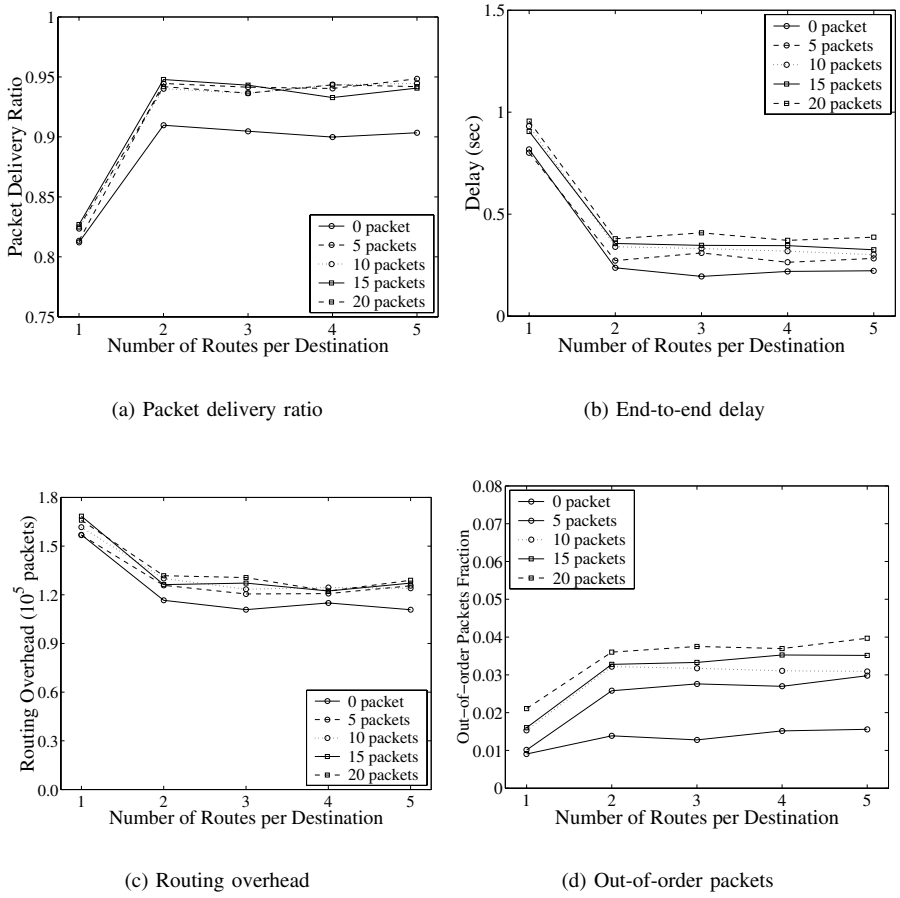
Node i deletes k from $\mathbf{S}^i_j$, where j is the destination of the packet. If i originated the referred packet and it has no other route j, it initiates a new route discovery if there is none in progress. If i has other routes to j and it has a copy of the data packet in its data cache, it forwards the data packet according to the data forwarding rule (see Section III-C). If i has no other route to j and it has a copy of the data being referred to in its data cache, it removes the referred data packet from its data cache and adds the data packet header information in the RERR it created. If i does not have the packet in its data cache and it is the previous hop of the packet, i adds the data packet header information in the RERR it created. If after parsing the RERR node i fails to salvage one or more data packets, it broadcasts the RERR it created.

IV. SIMULATION MODEL

All simulation models are based on the simulator program *ns-2* [19] with the extensions from the Monarch Project [20]. The simulations are divided into two parts. The first part evaluates the effect of data cache size and number of stored routes per destination on the performance of CHAMP. The second part is a performance comparison between CHAMP (using the optimum parameters derived in the first part), AODV and DSR.

The network used for the simulations consists of 100 nodes in a 1500 m $\times$ 600 m area. The movement of the nodes follows the ''random way-point'' model [9]. Six different pause times are used: 0, 30, 60, 120, 300, and 600 seconds. A zero pause time indicates that nodes are continuously moving while a 600-second pause time means that nodes are at rest for the entire simulation duration. Nodes move at speeds between 0 and 20 m/s.

For simulations using constant-bit-rate (CBR) sources, the packet size is set to 512 bytes and sending rate is set to 4



(a) Packet delivery ratio

(b) End-to-end delay

(c) Routing overhead

(d) Out-of-order packets

Fig. 1. Performance of CHAMP as a function of the number of routes per destination. Each line plot represents a particular value for the data cache size. The text in the legend refers to the size of the data cache.

packets per second. For simulations using TCP Reno sources, the packet size is set to 1460 bytes and maximum window to 8.

The two-ray ground reflection approximation is used as the radio propagation model. The wireless interface device is modeled after the 2 Mb/s Lucent WaveLan card [21]. For the medium access control protocol, the IEEE 802.11 Distributed Coordination Function (DCF) is used. The interface queue is a 50-packet drop-tail priority queue. At the network layer, a send buffer that can accommodate at most 64 packets is used for storing packets waiting for routes.

The performance of the protocols are evaluated using the following criteria. For the CBR simulations, the performance metrics are: (i) *packet delivery ratio* - the total number of packets delivered divided by the total number of packets sent; (ii) *end-to-end delay* - the delay for every packet delivered; and (iii) *routing overhead* - the total number of routing packets sent and forwarded throughout the simulations. For the TCP simulations, the performance metrics are: (i) *TCP throughput*, and (ii) *routing overhead* as defined previously. In evaluating CHAMP, we also measure the *fraction of packets that arrive out-of-order*.

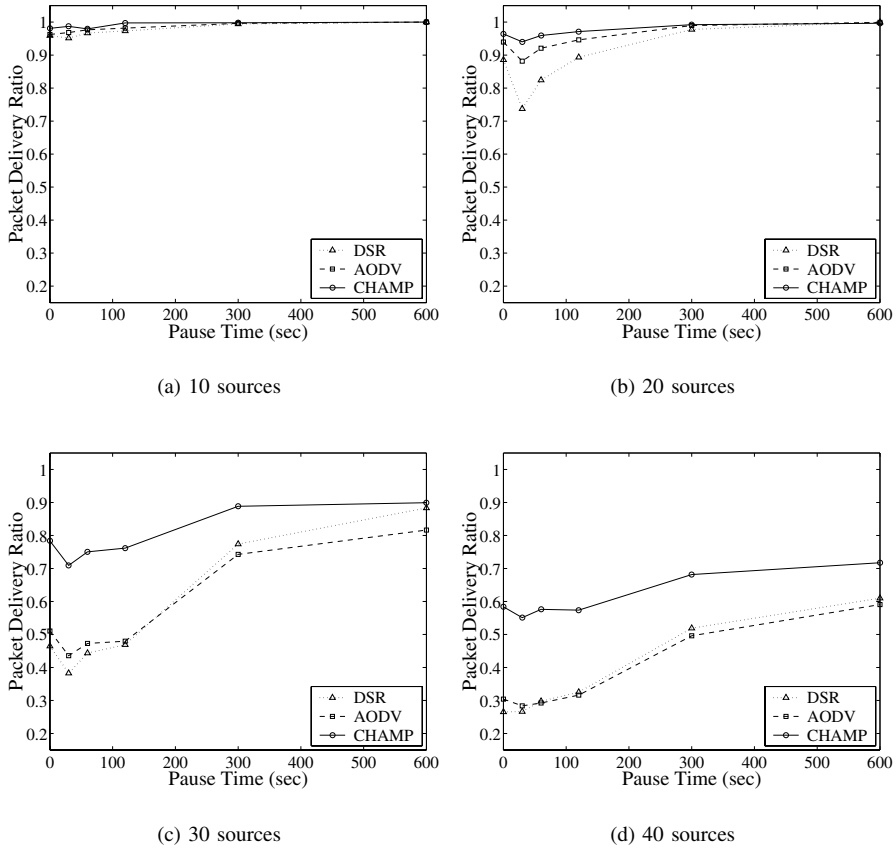
V. CHAMP Performance

The performance of CHAMP is influenced by two parameters, namely, *data cache size* and *number of stored routes per active destination*. We vary these parameters and determine their effects on the packet delivery ratio, end-to-end delay, routing overhead, and fraction of out-of-order packets. Fig. 1 shows the results when where there are 20 CBR sources at 0 pause time.

A. Impact of Data Cache Size

Every node that uses CHAMP maintains exactly one first-in first-out data cache regardless of the number of connections or destinations that node serves. A larger data cache means that more data packets can be stored at any given time. Since the data cache uses a simple first-in first-out replacement policy, a larger data cache also implies that data packets can stay longer in the cache. Increasing the data cache size therefore increases the probability of success of packet salvaging. Expectedly, the packet delivery ratio should incrementally increase as the data cache size is incrementally increased. However, the results do not completely support this hypothesis.

While the packet delivery ratio increases as the data cache



(a) 10 sources

(b) 20 sources

(c) 30 sources

(d) 40 sources

Fig. 2. Fraction of successfully delivered data packets as a function of mobility.

size is increased, the rise is only significant, around 4-5%, when the data cache size changes from 0 to 5. The packet delivery ratio for data cache sizes 5, 10, 15 and 20 are almost the same at 95%. This result suggests that there is no benefit in having a large data cache. Indeed, when we examined the trace files, we found out that due to temporal locality of dropped packets, only the most recent packets are used for packet salvaging. No matter how large the data cache is, the relatively old packets are never utilized; hence, they do not contribute to the packet delivery ratio.

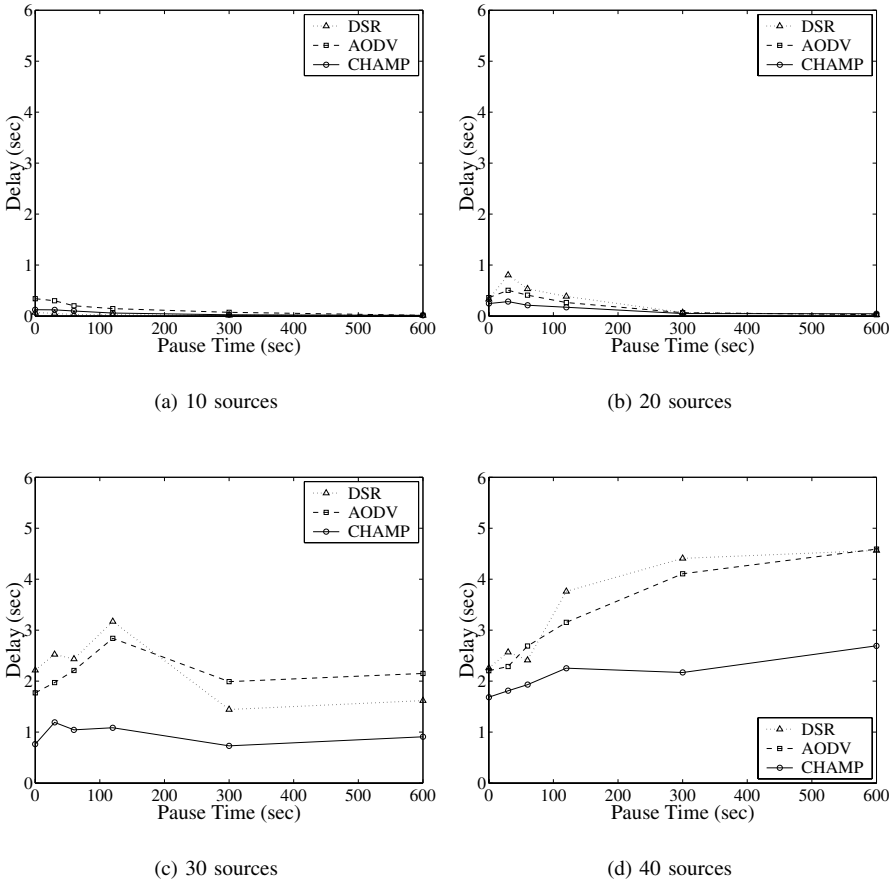
There is a slight increase in end-to-end delay as the data cache size is incrementally increased. Meanwhile, the fraction of out-of-order packets shows a significant two-fold rise as the data cache size increases from 0 to 5, and continues to increase from hereon. When the data cache size is 20 packets, the fraction of out-of-order packets becomes three-fold. The worse delay and out-of-order packet arrival when the data cache size is larger is expected. If the data cache is small, only nodes close to the route failure may have the packet in their respective caches. On the other hand, if the data cache is large, many nodes, including those that are far from the route failure (and therefore farther from the destination) may still have the packet in their respective caches. Consequently, if these far nodes can successfully salvage packets, delays should increase. Furthermore, the variation of these delays becomes bigger since both near and far nodes can salvage packets, leading to increased number of out-of-order packets.

B. Impact of Number of Stored Routes

The success of packet salvaging from data cache depends on the availability of at least one alternative route to the destination of a packet. Having the pertinent packet in the data cache does not guarantee that a salvaged packet will reach the destination. Logically, having more routes should result in increased packet delivery ratio since there will be a greater chance for salvaged packets to get delivered to the destination.

As the number of routes increases from one to two, the packet delivery ratio increases by 10% from 81% to 91%. There is a negligible increase as the number of stored routes is further increased to five. This indicates that there is no advantage in storing more than two routes for each destination. We found two reasons for this behavior. First, note that the parameter specifies the maximum value. Most nodes do not actually reach this value especially when it is high. And second, a node requires only one alternative route at any given time to successfully perform packet salvaging.

The presence of at least two routes significantly reduces both the end-to-end delay and routing overhead. Note that the end-to-end delay drops to almost half when the number



(a) 10 sources

(b) 20 sources

(c) 30 sources

(d) 40 sources

Fig. 3. End-to-end delay as a function of mobility.

of routes is increased to two or more. This benefit can be attributed to the per packet, round-robin load distribution feature of CHAMP. Meanwhile, the routing overhead decreases by 1/4 when there are two or more routes. This reduction is also expected since the presence of more than one route helps nodes to reduce route discoveries.

There is an increase in the fraction of out-of-order packets as the number of next hop choices increases from one to two. However, no further increase can be observed when the number of routes is increased to five. The increase when the number of routes increases from one to two is due to the per packet load distribution employed by CHAMP. But since CHAMP uses multiple routes of equal lengths, the fraction of out-of-order packets when there are two or more routes is almost the same.

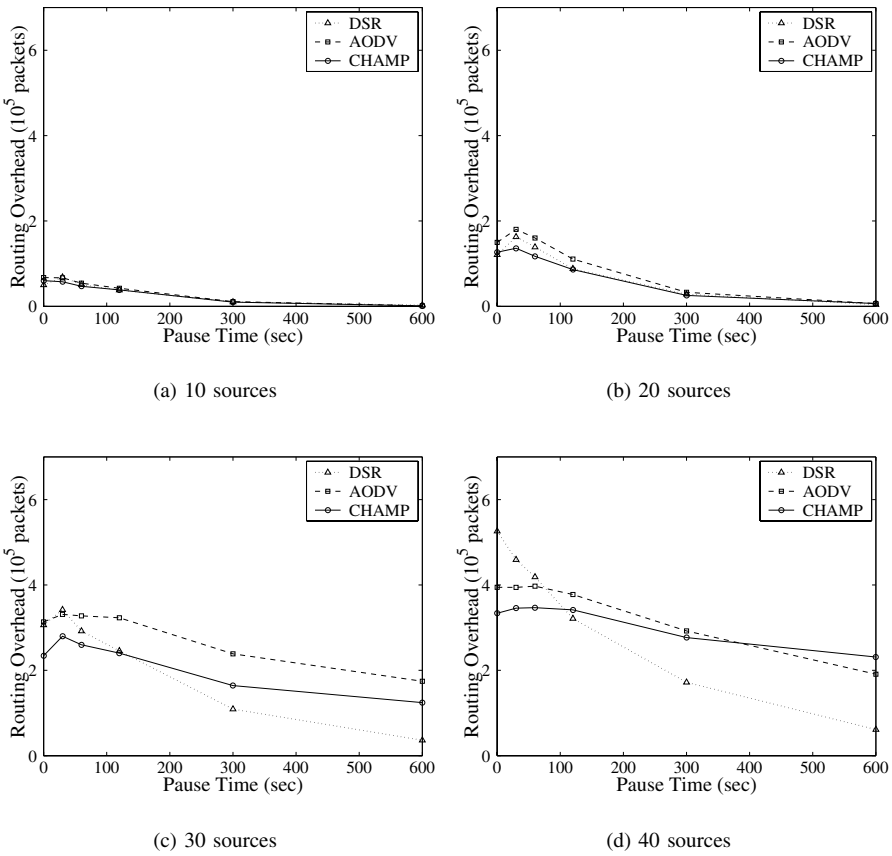
VI. Performance Comparison

In the previous Section, we have shown that cooperative packet caching and shortest multipath routing indeed improves the performance of CHAMP. Having a data cache that can accommodate at most five packets and storing two routes to every active destination already results in optimum performance. In this Section, we compare CHAMP, using a five-packet data cache and two routes per destination configuration, with DSR and AODV. The default protocol parameter values for AODV and DSR reported in a performance evaluation [9] are used as they offer the best performance. The simulations are divided into two sets. In the first set, source nodes are constant-bit-rate (CBR) sources while in the second set; source nodes are TCP Reno sources.

A. CBR Simulation Results

Fig. 2 shows the packet delivery ratio with varying number of CBR sources and averaged over 20 trials. CHAMP shows the highest packet delivery ratio especially in more mobile and higher load scenarios. Its biggest advantage occurs in the 30- and 40-source scenarios where it leads AODV and DSR by at most 30% at 0 pause time.

The packet delivery ratio for all the protocols decreases as the rate of mobility increases. The largest decrease in this metric can be observed in the 30- and 40-source scenarios. DSR incurs the largest drop of more than 45%. AODV decreases by at most 35% while CHAMP drops by not more than 20%. At 20 sources, DSR already shows a noticeable decrease of more than 25%. The low packet delivery of DSR can be attributed to its aggressive route caching without any



(a) 10 sources

(b) 20 sources

(c) 30 sources

(d) 40 sources

Fig. 4. Routing overhead in terms of packets as a function of mobility.

expiration mechanism. As we have observed and as noted in a related work [22], this policy causes stale routes to be used for packet relay causing additional bandwidth consumption due to mis-routed packets. More seriously, since any node can reply to route requests, stale routes contained in route replies cause further pollution of the route caches of other nodes.

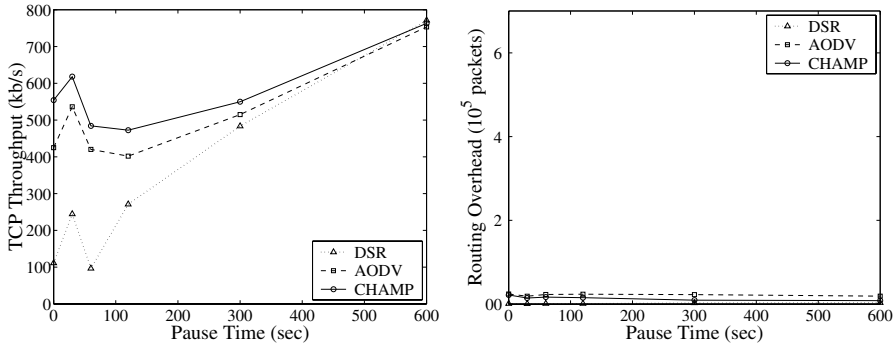
Packet delivery ratio is also affected by the number of sources. As expected, the packet delivery ratio decreases as the traffic increases. DSR and AODV show a significant decrease of 70% while CHAMP drops by 45% drop. The biggest change can be seen between the 20-source and 30-source scenarios. This significant drop can be attributed to network congestion at 30 sources. Note that even when there is no node movement, all protocols deliver less than 90% of the data packets. At 40 sources, the congestion becomes worse as the packet delivery ratio of all the protocols drops further.

A closer inspection of the trace files reveals the undesirable impact of congestion on the routing protocols. In a congested network, the medium access control protocol encounters many collisions when sending a packet. This latency in accessing the channel causes the interface queue to become full, causing packets to be dropped. Furthermore, if congestion is severe, it causes the MAC protocol to timeout. The MAC then informs the routing protocol that the packet cannot be forwarded to the next hop. Since the routing protocol is not aware of congestion, it assumes that a forwarding failure is due to link failure even when this is not the case. Consequently, the routing protocol either propagates route error messages or performs a new route discovery that causes additional routing overhead. This kind of reaction by the routing protocol worsens the congestion, causing more packets to be dropped and increasing the delay.

Fig. 3 shows the end-to-end delay for the simulations with varying number of CBR sources and averaged over 20 trials. CHAMP exhibits the smallest and most consistent end-to-end delay. Its largest performance advantage occurs in the 30- and 40-source scenarios where its delay is less than half that of AODV and DSR at certain mobility rates.

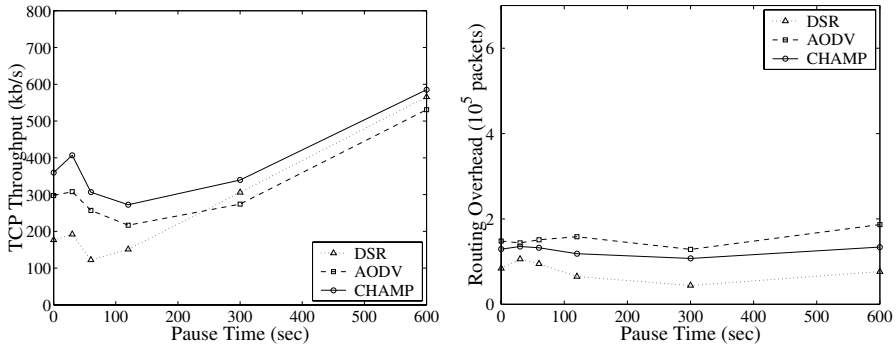
The end-to-end delay of all the protocols increases as the number of sources increases. At 10 and 20 sources, the delay of all the protocols is almost the same. At 30 sources, the delay of AODV and DSR shoots up to more than two seconds while the delay of CHAMP rises to more than one second. This dramatic increase is due to congestion. At 40 sources, the worsening congestion causes the delay of all the protocols to further increase. At this point, the delay of AODV and DSR rises to 4.5 seconds while the delay of CHAMP reaches 2.8 seconds.

One interesting observation at 40 sources is that the delay



(a) TCP Throughput (b) Routing overhead

Fig. 5. Simulation results for 1 TCP Reno connection.



(a) TCP Throughput (b) Routing overhead

Fig. 6. Simulation results for 1 TCP Reno connection with 10 CBR background traffic.

of all the protocols unexpectedly rises as the rate of mobility decreases. This result implies that mobility can somehow ease congestion, as delays are smaller at more mobile scenarios. When we examined the trace files, we found out that this is indeed the case. Node mobility aids in distributing traffic due to source movements, thereby reducing congestion in specific areas. This kind of phenomenon was also observed by prior simulation studies [22], [11].

Fig. 4 shows the routing overhead, in terms of packets, generated throughout the entire simulations with varying number of CBR sources and averaged over 20 trials. In terms of this metric, the performance of CHAMP is still comparable to that of AODV and DSR. While there is no significant improvement, these results indicate an excellent achievement for CHAMP since it is a multiple path routing protocol. Expectedly, CHAMP should generate a larger routing overhead since more nodes are involved in the route discovery particularly in the route reply propagation. However, note that storing multiple paths to active destinations reduces the need for route discoveries, and therefore routing overhead, since CHAMP only performs a new route discovery when it loses all its routes.

The routing overhead of all the protocols is affected by the number of sources. At 10 sources, all protocols generate less than 60K packets of overhead. At 20 sources, the routing overhead of AODV and DSR increases more than two-fold to around 150K packets at the pause time of 30 seconds. The routing overhead of CHAMP doubles to around 120K packets. At 30 sources, the routing overhead of all the protocols continues to increase two-fold. Finally at 40 sources, the protocols show their worst overhead. CHAMP generates at most 350K packets; AODV generates at most 400K packets and DSR at most 540K packets.

The routing overhead of the protocols is also sensitive to mobility. The observable trend is for the routing overhead to rise as the rate of mobility rises. DSR shows the biggest change at 40 sources as its routing overhead increases nine-fold from 60K to 540K packets. AODV and CHAMP show a more than two-fold increase at 30 and 40 sources. An interesting feature of the plots for the 30- and 40-source scenarios is that at low mobility rates (pause times of 600 and 300 seconds), DSR shows the lowest routing overhead. This

is the advantage of caching of overheard routes employed by DSR. At low mobility rates, since routes tend to be valid for longer periods, caching of overheard routes reduces the need for route discoveries. However at higher mobility rates, since routes are short-lived, cached routes are often invalid causing complications such as incorrect forwarding of packets to stale routes and pollution of route caches of other nodes.

B. TCP Simulations

As mentioned in Section III, CHAMP performs load-balancing on a per-packet basis that can possibly re-order the arrival of packets at the destination. To determine whether this can degrade TCP performance, we performed simulations using one TCP Reno connection. Figs. 5 and 6 show the TCP throughput and routing overhead in terms of packets for these simulations averaged over 20 trials.

In both tests, CHAMP shows the highest TCP throughput at all pause times. At higher mobility rates (less than 300 seconds of pause time), CHAMP outperforms AODV by 20% and DSR by 70%. In terms of routing overhead, all the protocols generate very small overhead when there is only one TCP connection. In the presence of the background traffic, AODV shows larger overhead than CHAMP and DSR. DSR shows the smallest routing overhead because of caching of overheard routes. However, this comes at the expense of lower TCP throughput.

The higher TCP throughput of CHAMP when mobility is higher can be attributed to its per-packet load balancing policy. In Fig. 1(b), we have seen that by using two or more routes, CHAMP reduces its end-to-end delay by 1/4. Furthermore, the use of equal-length routes do not significantly reorder the arrival of packets as shown in Fig. 1(d). In the performance comparison using CBR sources, CHAMP shows the smallest end-to-end delay at lower pause times. This translates to smaller round-trip times, and hence, higher TCP throughput for CHAMP at lower pause times.

We noticed that in some trials, DSR failed to send any TCP packet (0 throughput) while AODV and CHAMP both registered non-zero throughputs. This behavior has been reported in previous work [23] and can be attributed to the aggressive route caching and no route expiration policy of DSR. Invalid stale routes cause severe adverse effects on the TCP.

VII. Related Work

Since the introduction by Wilkes [12] in 1965, caching is now used in many systems such as distributed file systems and the World Wide Web and in various system layers [14]. In this paper, we introduced data packet caching in the context of the mobile ad hoc networks. Performance improvements in terms of higher packet delivery, lower delays and reduced routing overhead are obtained due to the temporal locality of dropped packets. A related work by Castaneda and Das [15] investigated the spatial locality of mobile nodes and observed that ''a mobile node cannot move too far too soon''. They exploited this property to lower routing overhead during route discovery and repair.

One of the earliest works on adaptive multipath routing, proposed by Gafni and Bertsekas [24], uses a series of ''link-reversals'' to form a directed acyclic graph (DAG) rooted at the destination. One problem is that this protocol, which came to be known as GB, exhibits instability when the network is partitioned. Corson and Ephremides [25] developed a new algorithm based on this concept, termed Lightweight Mobile Routing (LMR) algorithm. Subsequently, the same authors introduced TORA [3]. LMR and TORA share many similarities including the route construction and route maintenance phases. TORA has an additional phase known as route deletion. One major drawback of TORA is the maintenance overhead. CHAMP eliminates such overhead by spreading packets over all the routes in a round-robin fashion.

Vutukury and Garcia-Luna-Aceves [16] used shortest multipath routes in MDVA, a proactive multipath distance-vector routing protocol for a network with changing topology. The emphasis of their work is to ensure loop-freedom and correctness at every instant by using loop-free invariant conditions. MDVA operates proactively and may not be suitable for mobile ad hoc networks. In this study, we developed a fast and simple way of discovering shortest multipath routes that does not cause large overheads.

Several studies have made multipath extensions to single-path routing protocols. Lee and Gerla [26] proposed an extension to AODV, called AODV-BR (AODV with back-up routes). AODV-BR utilizes multiple routes organized in a ''fish-bone'' structure and has been shown to improve protocol performance and robustness against mobility in light load conditions. However, unlike CHAMP, it does not perform well in highly loaded scenarios.

Marina and Das [27] proposed another multipath extension of AODV called AOMDV. It uses the notion of an ''advertised hop count'' to maintain multiple loop-free paths. This approach also lends some similarities to CHAMP's route discovery procedure. Simulation results show that AOMDV outperforms AODV by as much as 5% in terms of packet delivery, half the delay and 20% less routing overhead. While CHAMP discovers non-disjoint multipath routes, AOMDV ensures the discovery of link-disjoint routes. A recent study have shown that non-disjoint multipath routes are more resilient and energy-efficient than disjoint routes [28].

Nasipuri and Das [29] proposed a multipath extension to DSR. As noted in many similar studies [22], [23], DSR's aggressive route caching and no route expiration policy cause severe adverse effects on both CBR and TCP connections in mobile scenarios. Hence, extending DSR to support multiple routes without addressing the caching problem may also multiply this problem. Furthermore, there is a high probability that the alternative route is always stale, as it is never used until the primary route fails. CHAMP cleverly avoids this problem by using all the available routes in a round-robin fashion.

Gallager [30] introduced an algorithm for distributing load over multiple paths that leads to minimum delays. As the algorithm converges slowly, it is unsuitable or networks

where load is highly dynamic. Vutukury and Garcia-Luna-Aceves [31] discussed an approximate solution to the problem. Krishnan and Silvester [32] found that a per packet load distribution policy provides the best results. However, this comes at the expense of out-of-order packet delivery, which can incur additional packet re-sequencing delays [33] and may adversely affect TCP connections [34]. CHAMP employs a simple load-balancing algorithm to distribute the load. Packets are spread over all the routes in a round-robin fashion. Since chosen routes are of equal length, packet reordering is not severe.

VIII. Conclusions

In this paper, we introduced cooperative packet caching, a strategy similar to cooperative caching aimed at reducing packet loss due to frequent route breakdowns. We have shown that because of the property of temporal locality in dropped packets, a small data cache (five packets) is sufficient to improve packet delivery. In essence, cooperative packet caching enables distributed packet salvaging. Although caching is now employed in many distributed systems, this is the first application in the network layer, particularly of mobile ad hoc networks.

A multiple path route discovery mechanism suitable for mobile ad hoc networks is also developed. The discovery mechanism selects non-disjoint shortest multipath routes of equal distance. We have shown that having at most two routes for every destination is already the optimum. Having more routes does not give any additional benefit. Obviously, a five-packet data cache and two routes per destination configuration does not entail a large overhead.

We have shown that by using the above-mentioned values for the data cache size and number of stored routes per destination, significant improvement can be obtained over AODV and DSR especially in more dynamic and higher load scenarios. In terms of packet delivery, CHAMP outperforms AODV and DSR by at most 30%. In highly congested scenarios, the delay of CHAMP is half that of AODV and DSR. In terms of routing overhead, CHAMP generates a relatively lower overhead at higher mobility rates. Although CHAMP has a higher percentage of out-of-order packets due load-balancing in a per-packet basis, it does not degrade TCP performance.

References

[1] M.S. Corson and J. Macker, ''Rfc 2501: Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations,'' Jan. 1999.

[2] C. Perkins and P. Bhagwat, ''Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers,'' in *Proc. ACM SIGCOMM '94*, Aug. 1994, pp. 234--244.

[3] M.S. Corson and A. Ephremides, ''A distributed routing algorithm for mobile wireless networks,'' *ACM/Baltzer Wireless Networks J.*, vol. 1, no. 1, pp. 61--82, Feb. 1995.

[4] S. Murthy and J.J. Garcia-Luna-Aceves, ''An efficient routing protocol for wireless networks,'' *Mobile Networks and Applications*, vol. 1, no. 2, pp. 183--197, 1996.

[5] R. Dube, C. Rais, K.Y. Wang, and S.K. Tripathi, ''Signal stability-based adaptive routing (ssa) for ad hoc mobile networks,'' *IEEE Personal Commun.*, vol. 4, no. 1, pp. 36--45, Feb. 1997.

[6] D. Johnson and D. Maltz, ''Dynamic source routing in ad hoc wireless networks,'' in *Mobile Computing*, T. Imielinski and H. Korth, Eds., pp. 153--181. Kluwer Academic Publishers, Norwell, Mass., 1996.

[7] Z. Haas, ''A new routing protocol for the reconfigurable wireless networks,'' in *Proc. IEEE ICUPC '97*, Oct. 1997.

[8] C. Perkins and E. Royer, ''Ad hoc on-demand distance vector routing,'' in *Proc. IEEE WMCSA '99*, Feb. 1999, pp. 90--100.

[9] J. Broch, D. Maltz, D. Johnson, Y.C. Hu, and J. Jetcheva, ''A performance comparison of multi-hop wireless ad hoc network routing protocols,'' in *Proc. ACM/IEEE MOBICOM '98*, Oct. 1998, pp. 85--97.

[10] S.J. Lee, C.K. Toh, and M. Gerla, ''Performance evaluation of table-driven and on-demand ad hoc routing protocols,'' in *Proc. IEEE PIMRC '99*, Sept. 1999, pp. 297--301.

[11] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, ''Scenario-based performance analysis of routing protocols for mobile ad-hoc networks,'' in *Proc. ACM/IEEE MOBICOM '99*, 1999, p. 195–206.

[12] M.V. Wilkes, ''Slave memories and dynamic storage allocation,'' in *Trans. IEEE*, Apr. 1965, vol. EC-14, pp. 270--271.

[13] D. Patterson and J. Hennessy, *Computer Architecture, A Quantitative Approach*, Morgan Kaufmann, 1996.

[14] V. Milutinovic, ''Caching in distributed systems,'' *IEEE Concurrency*, pp. 2--3, July-September 2000.

[15] R. Castaneda and S. Das, ''Query localization techniques for on-demand routing protocols in ad hoc networks,'' in *Proc. ACM/IEEE MOBICOM '99*, 1999, pp. 186--194.

[16] S. Vutukury and J.J. Garcia-Luna-Aceves, ''Mdva: a distance-vector multipath routing protocol,'' in *Proc. IEEE INFOCOM '01*, 2001, vol. 1, pp. 557--564.

[17] A. Valera, K.G. Seah, and S.V. Rao, ''Champ: A highly resilient and energy-efficient routing protocol for mobile ad hoc networks,'' in *Proc. IEEE MWCN '02*, 2002, pp. 43--47.

[18] J.J. Garcia-Luna-Aceves, ''Loop-free routing using diffusing computations,'' *IEEE/ACM Trans. Networking*, vol. 1, no. 1, pp. 130--141, 1993.

[19] The VINT Project, ''The network simulator - ns-2,'' Available at http://www.isi.edu/nsnam/ns/.

[20] The Monarch Project, ''Rice monarch project: Mobile networking architectures,'' Available at http://www.monarch.cs.rice.edu/.

[21] B. Tuch, ''Development of wavelan, an ism band wireless lan,'' *AT&T Technical Journal*, vol. 72, no. 4, pp. 27--33, 1993.

[22] S. Das, C. Perkins, and E. Royer, ''Performance comparison of two on-demand routing protocols for ad hoc networks,'' in *Proc. IEEE INFOCOM '00*, 2000, pp. 3--12.

[23] G. Holland and N. Vaidya, ''Analysis of tcp performance over mobile ad hoc networks,'' in *Proc. ACM/IEEE MOBICOM '99*, Aug. 1999, pp. 219--230.

[24] E. Gafni and D. Bertsekas, ''Distributed algorithms for generating loop-free routes in networks with frequently changing topology,'' *IEEE Trans. Commun.*, vol. 29, no. 1, pp. 11--15, January 1981.

[25] M.S. Corson and A. Ephremides, ''A distributed routing algorithm for mobile radio networks,'' in *Proc. IEEE MILCOM '89*, Oct. 1989.

[26] S. Lee and M. Gerla, ''Aodv-br: Backup routing in ad hoc networks,'' in *Proc. IEEE WCNC '00*, Sept. 2000.

[27] M. Marina and S. Das, ''On-demand multipath distance vector routing in ad hoc networks,'' in *Proc. IEEE ICNP '01*, 2001, pp. 14--23.

[28] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, ''Highly-resilient, energy-efficient multipath routing in wireless sensor networks,'' in *Proc. ACM MOBIHOC '01*, Oct. 2001, pp. 251--253.

[29] A. Nasipuri and S. Das, ''On-demand multipath routing for mobile ad hoc networks,'' in *Proc. IEEE ICCCN '99*, Oct. 1999, pp. 64--70.

[30] R. Gallager, ''A minimum delay routing algorithm using distributed computation,'' *IEEE Trans. Commun.*, vol. 25, pp. 73--84, Jan. 1977.

[31] S. Vutukury and J.J. Garcia-Luna-Aceves, ''A simple approximation to minimum-delay routing,'' in *Proc. SIGCOMM '99*, 1999, pp. 227--238.

[32] R. Krishnan and J. Silvester, ''Choice of allocation granularity in multipath source routing schemes,'' in *Proc. IEEE INFOCOM '93*, Mar. 1993, pp. 322--329.

[33] N. Gogate and S. Panwar, ''Assigning customers to two parallel servers with resequencing,'' *IEEE Trans. Commun. Lett.*, vol. 3, no. 4, pp. 119, Apr. 1999.

[34] N. Gogate and S. Panwar, ''Supporting applications in a mobile multihop radio environment using route diversity,'' in *IEEE ICC '98*, June 1998.