TrAM: Cross-Layer Efficient Application-Layer Multicast in Mobile Ad-hoc Networks

Peter Baumung
Institute of Telematics
Universität Karlsruhe (TH)
Email: baumung@tm.uka.de

***Abstract*— Application-layer multicast protocols more and more appear as attractive multicasting approaches, as they combine increased customizability of protocol mechanisms with the latters' ease of deployment. Both features especially become important in mobile ad-hoc networks, in which groups of users running applications with diverging requirements share limited network resources. The core of each application-layer multicast protocol is the so-called overlay network. Since the latter is used for data forwarding between group members, its topology as well as its topology maintenance mechanisms have a direct impact on the communication's efficiency. Their design thus requires special attention and detailed evaluations, with respect to potentially involved cross-layer effects. In this contribution we propose a novel overlay topology, which, because of its lightweight design, features an increased performance. For evaluation, we rely on simulative cross-layer analysis, in which we compare our protocol to other application-layer multicast approaches as well as to a simple *n*-times unicast strategy.**

I. Introduction

Mobile ad-hoc networks (MANETs) consist of mobile devices, that communicate via the wireless medium without any fixed infrastructure. While two devices located in one another's transmission range communicate directly, intermediate devices bridge distances between farther nodes. As multi-hop communication is enabled, a complex and potentially dynamic wireless network arises. Here, many application share their need for multicast communication. Indeed, cooperating and thus communicating groups will be present in the majority of educational, touristic, rescuing or military scenarios.

From the success of *Peer-to-Peer* applications in the fixed Internet arose the idea of moving multicast functionality to the application-layer of user devices. There, packet duplication and group management are handled by the multicast group members' themselves. Data packets are forwarded using a so-called *overlay network*, which consists of unicast tunnels interconnecting the group's members. Multicast functionality thereby becomes easily deployable, since the underlying network is required to provide support for unicast routing only. As it used for forwarding data packets, the overlay network has a direct impact on the efficiency of application-layer multicast protocols. Its topology is required to be maintained through time, since the multicast group as well as the underlying (physical) network can show a dynamic character: Indeed, while user devices may join or leave a group at any time, they also can show a certain degree of mobility, and thus cause unicast tunnels to worsen with respect to a given metric. Topology maintenance together with the propagation of current (overlay) routing information involves a certain amount of control flow. Because bandwidth is a scarce resource in MANETs, the design and the evaluation of an overlay's topology thus requires dedicated attention.

For the analysis of protocols, previous contributions mainly refer to standard values, such as control flow measurements on the application-layer, an overlay's average link cost and achieved delivery ratios. We however argue, that these kind of measurements, by themselves, are not meaningful in environments such as MANETs. In order to truly rate an overlay's efficiency, it here becomes a key factor to analyze potential cross-layer effects of the traffic emitted on the application-layer. Indeed, when e.g. using reactive routing protocols such as AODV [1], setting up unicast tunnels can involve expensive route discoveries on the network-layer. We thus dedicate this work to the design and evaluation of an overlay topology that minimizes cross-layer effects. To rate its efficiency, we compare our protocol to existing solutions by directly measuring the load induced by the different protocols on the MANET's physical layer.

The remainder of this paper is organized as follows. Section II briefly presents Narada and PAST-DM, two previously developed application-layer multicast protocols. In section III, we introduce TrAM, a novel overlay topology, which has been optimized with respect to cross-layer effects. Section IV compares TrAM to Narada and PAST-DM as well as to a simple *n*-times unicast strategy, in the context of a standard CBR and a chat application. Eventually, section V concludes by giving a short summary and an overview of topics that will be handled in future work.

II. Related Work

Although initially developed for the fixed Internet, Narada [2] shows to be a promising application-layer multicast protocol for MANETs. Indeed, Narada relies on a flexible mesh-like topology and includes basic topology maintenance operations. The latter let each group member periodically probe the link quality to all other group members. When considered as efficient, the link is included in the topology and the latter's routing. Accordingly, when considered as inefficient, an existing link is shut down. To prevent the overlay's partitioning, the protocol includes special mechanisms to enforce overlay links to distant (and thus sub-optimal) nodes. Routing itself

is based on a reverse shortest path algorithm, requiring the periodic exchange of link information between neighboring nodes.

In contrary to Narada, PAST-DM [3] was developed for MANETs. It relies on a link-state source-routing algorithm, which requires all group members to have a full view on the overlay's topology. This is achieved by letting each group member periodically send all its known links states to its neighboring nodes. Depending on the multicast group's size, the propagation of links states thus involves a potentially increased control flow overhead. As, additionally, link-states are propagated on a periodic base, the freshness of routing information used by forwarding group members and multicast sources is questionable. The protocol's routing is thus likely to become unstable in case of changes inside the overlay's topology. The latter can occur, as group members periodically look for better or worsening neighbors in their vicinity.

III. TrAM - A Cross-Layer Efficient Overlay

In this section we introduce TrAM, a novel overlay topology. Its main design goals were to have little control flow with respect to cross-layer effects, as well as a stable, flexible and lightweight overlay topology, that takes a MANET's dynamics into account. TrAM acquires its lightweight character by avoiding any redundant unicast tunnels within its topology. It thus organizes the multicast group as a logical tree structure, in which any of the tree's members only know their successors (children) and predecessor (parent) with respect of the tree's root node. This structure makes data delivery very easy, as sending and forwarding nodes only need to transmit data packets to their neighbors.

A. Tree Construction and Group Join

An initial condition to make the TrAM protocol work properly is the existence of the tree's root node. We let the first joining node automatically create the multicast group by declaring itself as the root node. Every node, that has successfully joined the group, can reach this root node (possibly via multiple overlay hops) through its parent node. To construct TrAM's tree, any node wanting to join the group has thus to find an existing group member that can work as its parent. To do so, every joining node uses a parent discovery process, during which it broadcasts so-called QueryParent packets. As broadcasts have a limited time to live (TTL), the latter is exponentially increased in case no reply from existing group members is received. Group members receiving a QueryParent packet can advertise themselves as a parent node, by answering with a ParentAdvertise packet. The joining node collects these answers for a short period of time and then chooses the most suitable node as parent.

In order to estimate the connection quality to potential parent nodes by objective criteria, a metric is required. For the latter, TrAM uses two parameters: The layer-3 hop count between the child and its parent on the one hand, and the number of overlay hops from the parent to the root node on the other hand. While for optimal connections both values

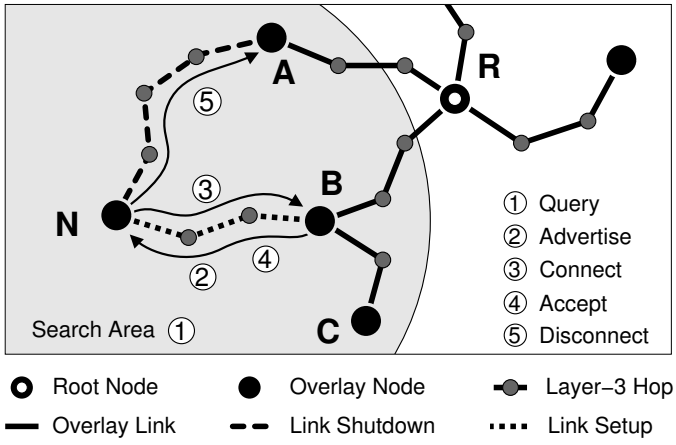

Fig. 1. The process of topology maintenance.

are tried to be minimized, the layer-3 hop count is however classified as more important. As this distance information is included in the ParentAdvertise messages, the joining node can decide through which parent it best connects to the multicast group. Group connection is done by registering at the chosen parent using a ConnectRequest packet. The parent may either accept or reject this request. In case of acceptation, the connection is confirmed with a ConnectAccept packet, after which the requesting node has successfully joined the group. The response contains further information about the constructed tree, namely the entire overlay hop path to the root node (PTR). With this knowledge, the newly joined node can advertise itself to other joining nodes.

B. Tree Maintenance

To make the protocol adaptive in dynamic wireless environments, the overlay's reconfiguration is an important issue. The distance between a node and its parent may increase, causing other nodes to become more suitable for maintaining a connection to the multicast group. For neighbor discovery, nodes use the same process as for initially joining the group. Nodes thus periodically broadcast QueryParent packets with limited TTL. This TTL is calculated from the layer-3 hop count to the parent increased by one. When connected to the multicast group, QueryParent packets include the current parent node's address, the latter's layer-3 hop distance as well as the PTR length and the depth of the node's subtree. Nodes receiving QueryParent messages react accordingly. The current parent node answers using a ParentHello in which the updated hop distance and the current PTR are included. Other group members receiving a QueryParent packet can, using the information included in the packet, decide whether they would be a better parent node for the emitting node. In this case, they respond with a ParentAdvertise packet, including their layer-3 hop distance and their PTR length. Nodes that recognize to be a worse parent to the emitting node, do not respond at all. Depending on advertisements arriving at the requesting node, the latter may either select a new parent, or stay connected to its current parent.

The exact process is shown in Figure 1, in which a node N currently is connected to the root R through node A. When looking for its optimal parent node, N in a first step broadcasts a QueryParent packet within the depicted search area[1]. Because of the information included in the query, node B recognizes to be a better parent for node N (since it lies one layer-3 hop closer to N than A) and, thus, advertises itself in a second step. As N gets aware of B, it decides to switch parent nodes. This is done, by first connecting to the new parent B and then, after the connection's acceptation, disconnecting from the former parent A. Node C remains quiet, as it, compared to N's current parent A, shows an increased PTR length and no improve in layer-3 hop distance.

Because of the metric described above, all nodes tend to align themselves to the root. In order to keep TrAM's average tree depth low and to improve the tree's quality, the protocol integrates a mechanism for redirecting root functionality. To do so, all group members include information about their subtree depth in their periodically sent QueryParent packets. Information about a subtree's depth is thus recursively propagated towards the root node. In case the latter detects a subtree that is more than two (overlay) hops deeper than any other subtree, the redirection process is initiated. When doing so, the root first sends a RedirectRequest packet to the preferred node, waiting for a RedirectAccept or, if the target is not willing to accept the redirection, a RedirectDeny.

C. Group Departure and Node Failure

When leaving the multicast group, a node unregisters at its parent and child nodes using a special Disconnect message. For reconnecting to the multicast group, the child nodes use the standard parent discovery process mentioned above. Whenever the root node wants to leave the multicast group, it, in first step, is required to successfully redirect its functionality to one of its child nodes.

Overlay-based protocols commonly get aware of node failures through consecutive losses of periodically exchanged heartbeat messages. As described above, TrAM nodes periodically query their parent node using QueryParent messages. In case of consecutively missing replies (ParentHello), a parent is declared as failed. The querying node thus reconnects to the group using the standard neighbor discovery process. In case of root node failure, its child nodes start the same reconnection procedure. As they connect to each other, all children, except one, will successfully locate a new parent node. After having reached a maximum threshold for its QueryParent's TTL, the remaining node has failed to reconnect to a root node and thus declares itself as the groups new root.

D. Discussion of TrAM's Features

Using the described neighbor discovery process, TrAM's topology maintenance is for each node reduced to a single broadcast with limited TTL and very few unicasts for potential advertisement replies. Unlike other protocols, TrAM does, one the one hand, not cause the underlying (layer-3) routing to set up many routes. While these are commonly required by application-layer multicast protocols to probe the link quality to potential neighbors, the process of route discovery however can be expensive, and its use should thus be minimized. On the other hand, TrAM, using its single broadcast, reaches a higher number of potential parents with constant overhead, resulting in an efficient and fast neighbor discovery mechanism.

Neighbor discovery plays an additional role considering TrAM's lightweight overlay topology. Unlike other protocols, TrAM avoids any redundant unicast tunnels between nodes. The latter usually exist in order to quickly adapt an overlay's routing in case of node failures. Failure detection however is a slow process, as it relies on consecutive losses of periodic heartbeat messages. While TrAM uses the same failure detection method, the routing's repairing is equivalent to locating a new neighbor using the standard node discovery process. Since the latter is comparatively faster than the former[2], TrAM, in case of node failures, performs only slightly worse than other overlay topologies. Because of its lacking redundancy, TrAM however saves an important amount of control flow overhead. The latter's extent becomes visible in the next section, in which TrAM is compared to other overlay topologies.

IV. Evaluation

In order to rate an overlay's performance, we, on the one hand, consider it as important, not to restrain evaluations to one single application scenario. We in this section thus compare TrAM to other overlays in the context of a single-source CBR and a multi-source chat application. On the other hand, we not only measure standard values, such as achieved delivery ratios and the observed latencies. To rate overlay topologies with respect to involved cross-layer effects, we additionally measure the *medium access time*. The latter sums the durations of medium accesses network-wide for each second of simulation time. This metric thus not only covers traffic emitted on the application-layer, but also monitors overhead generated by network routing and MAC[3].

We limit evaluations to scenarios with pedestrian mobility, ranging from 1 to $3\frac{m}{s}$. We let 30 nodes, that move according to RPGM [4] with a mean cluster size of 3 nodes and a radius of $50m$, join the multicast group within the first minute of simulation time. 90 additional nodes move according to the random direction mobility model. All nodes have a transmission range of $175m$ and roam on a surface of $1000m$ by $1000m$. While standard $2\frac{mbit}{s}$ IEEE 802.11b with its RTS/CTS extension is used as MAC, AODV provides unicast routing on the network-layer. All measurements are smoothed by averaging results of 20 different mobility scenarios and random seed values.

[1] Note that the ParentHello returned by N's current parent node, A, is omitted for the sake of clarity.

[2] As heartbeat periodicity is commonly set to $10s$, node failure detection often takes about $30s$ to $40s$, i.e. 3 or 4 consecutively lost heartbeats. TrAM's node discovery process, however, usually locates nodes in less than $1s$.

[3] Since accessing the medium is a highly energy consuming process, measuring the network-wide medium accesses also allows a protocol's rating with respect to its energy consumption. Although interesting, we do not further investigate this aspect in this contribution.

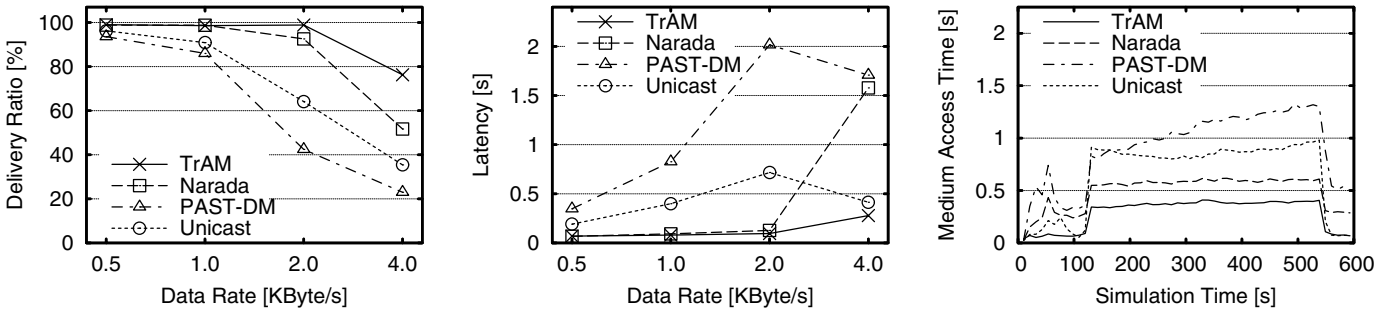

Fig. 2. Evaluation of a simulated, single-source CBR application.

A. Evaluating a Single-Source CBR Application

For evaluating a CBR application, we let a single source multicast data packets with a size of *512 bytes*. The source's traffic is varied, by increasing the number of packets sent per second. Figure 2 *left* shows the delivery ratios achieved by different overlay topologies. As can be seen, delivery ratios for TrAM remain at a constant high level for a data rate of up to 2 *kbyte/s* and drop only for a rate of 4 *kbyte/s*. While a slight drop of Narada's performance is already visible at 2 *kbyte/s*, results become unsatisfactory at 4 *kbyte/s*. The simple unicast strategy, which lets the source directly unicast packets to all group members, shows a worse performance than TrAM and Narada. This can be explained by its inefficient data forwarding, that overburdens the medium in the source's vicinity. Surprisingly, PAST-DM by far shows the worst performance of all investigated protocols. As will be shown below, its topology indeed involves too much overhead to cope with data rates beyond 1 *kbyte/s*.

Latencies are an interesting first indicator for a protocol's overhead, since packet delays mostly arise from collisions during packet transmissions and the MAC's exponentially growing back-off. Figure 2 *center* shows the latencies observed during simulation experiments. As can be seen, TrAM and Narada remain on an equally high level for a data rate of up to 2 *kbyte/s*. While at 4 *kbyte/s* TrAM's drop in performance remains acceptable (latencies stay below 0.3*s*), results for Narada drastically worsen to more than 1.5*s*. Because of too much overhead, latencies achieved by the unicast strategy and PAST-DM rapidly increase with the load offered by the source. Note that, at the highest data rate, latencies drop for the unicast strategy and PAST-DM. This directly results from the poor delivery ratios achieved by both protocols: As packet drops occur frequently, potentially highly delayed packet no longer affect latencies.

To investigate the actual overhead generated by the protocols more closely, we in figure 2 *right* plot the network-wide medium access time as defined above. Because of size constraints, we only provide measurements for the data rate of 1 *kbyte/s*, which we consider the most interesting. In the first 60*s* simulation time (plotted on the x-axis), 30 nodes join the multicast group. By doing so and depending on the used overlay topology, they generate a specific overhead. The latter becomes visible as peaks in the respective curves. As can be seen, in contrary to other overlay topologies, TrAM's overhead remains at a constant level. This can be explained by TrAM's highly efficient neighbor discovery process (c.f. section III-D), which is used for joining the multicast group as well as for topology maintenance. At 120*s* simulation time the multicast source starts emitting traffic. This becomes visible through the abrupt rise for all curves. The latter show that TrAM's topology involves the fewest overhead. Indeed, Narada causes the medium to be accessed about 50% longer than TrAM does. This can be explained through Narada's more complex topology which maintains redundant unicast tunnels and thus causes more overhead. The simple unicast strategy accesses the medium about twice as long as TrAM, which can be ascribed to an increased length of unicast tunnels. Indeed, since the source directly unicasts its packets to all group members, routes to receivers are longer than for a protocol which involves other group members in packet duplication and forwarding. While PAST-DM initially shows a performance comparable to the unicast strategy, medium accesses are increasingly required during data transmission. As the medium becomes heavily loaded, PAST-DM's overlay routing gets unstable and causes increasing overhead. At 540*s* simulation time traffic generation is stopped, resulting in dropping curves for all protocols. Interestingly, the remaining 60*s* simulation time do not suffice to let PAST-DM's topology recover from its instability during data delivery. Indeed, PAST-DM's medium access time does not drop to the same level as right before traffic generation is started.

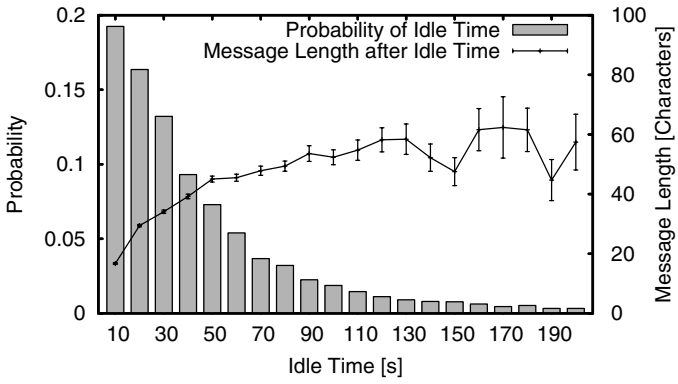

Fig. 3. Measurement of chat traffic.

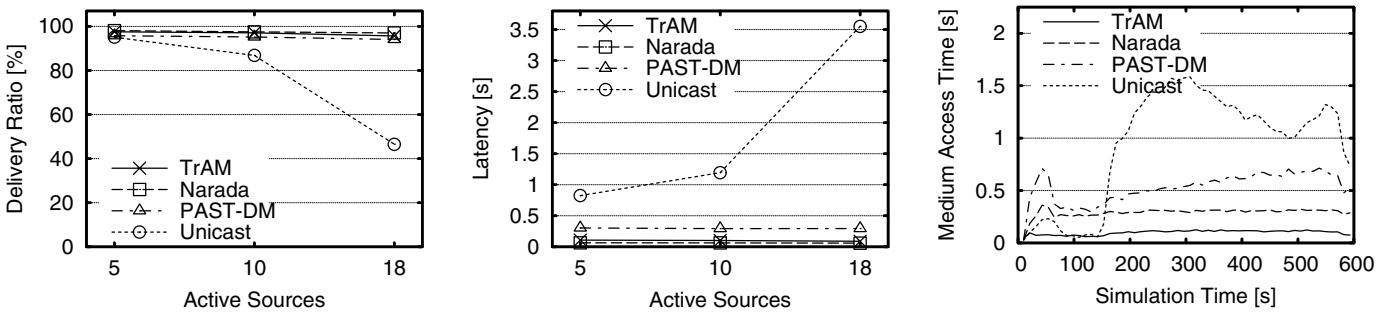

Fig. 4. Evaluation of a simulated, multi-source chat application.

B. Evaluating a Multi-Source Chat Application

For the realistic simulation of a chat application, an import aspect is the modeling of traffic. We therefore monitored conversations in typical chat rooms for a couple of days. The "average user behavior" resulting from the obtained measurements is shown in figure 3. The x-axis shows a user's *idle time*, which is the amount of time that elapses between two consecutive messages sent by the same user. While on the left y-axis the diagram shows the probability of a specific idle time, the right y-axis denotes the message length the user has generated within the respective idle time. Note that raw measurements show a much finer granularity and were condensed only for the sake of clarity.

We now oppose the different protocols to the traffic above and vary the number of active sources. The results of simulation experiments are shown in figure 4. As can be seen, delivery ratios (plotted in figure 4 *left*) are on a very high level (about 97%) for all (true) overlay multicast protocols. The simple unicast strategy, however, does not scale with the number of active sources. Indeed, as the number of routes set up by the network-layer grows quadratically, the protocol involves too much overhead. The latter also becomes visible in the measured latencies, which are shown in figure 4 *center*. The rapidly growing delays show that the unicast protocol overburdens the medium, resulting in frequent collisions and exponentially growing MAC back-offs. With latencies around $0.2s$ TrAM and Narada perform about equally well. PAST-DM also copes satisfactorily with chat traffic, and achieves latencies of $0.3s$.

While TrAM, Narada and PAST-DM show good results in terms of delivery ratios and latencies, the achieved performance comes at highly different cost. Figure 4 *right* shows the measured medium access time for 18 sources, which generate traffic between $120s$ and $540s$ simulation time. As can be seen, TrAM in terms of involved network load, out performs the other protocols. Indeed, while Narada accesses the medium about 3 times as long as TrAM, PAST-DM involves 6 times TrAM's network load. The diagram also shows the unicast strategy's increased overhead and its collapsing communication between $300s$ and $490s$ simulation time.

V. Conclusion and Future Work

In this contribution we proposed TrAM, a lightweight tree-based application-layer multicast protocol that was designed to minimize cross-layer effects. Using its unique topology maintenance process, the protocol avoids redundant unicast tunnels and thus saves an important amount of control flow overhead. To compare our protocol to previous studies, we use simulative cross-layer evaluations, during which we directly measure the medium access time required by all protocols. We can thus show that, in terms of involved network load, TrAM considerably outperforms existing solutions, such as Narada or PAST-DM. As a direct consequence, TrAM offers increased scalability in terms of data traffic emission. We underlay this by referring to measurements of standard values, such as achieved delivery ratios and latencies: Here, TrAM performs at least as good as existing solutions.

In future work we plan to further improve protocol scalability in terms of data traffic emission. Standard application-layer multicast protocols use unicast messages for data forwarding. Unicasts however suffer from scalability issues, especially in MANET scenarios with increased group member density. We thus plan to use limited broadcasts not only for the overlay's maintenance, but also for data delivery. As broadcasts however are not covered by MAC retransmissions, they are more fragile regarding radio interference. We will provide detailed evaluations to show the pros and cons of broadcasting data.

Acknowledgments

The authors would like to thank Björn Krämer, Denis Martin and Tobias Schlager for the valuable help in developing and implementing protocol mechanisms as well as conducting numerous simulation experiments.

References

[1] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das, "Ad hoc on-demand distance vector (AODV) routing," Feb. 2003.

[2] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS 2000*, Santa Clara, California, USA, June 2000.

[3] Chao Gui and Prasant Mohapatra, "Efficient overlay multicast for mobile ad hoc networks," in *The Wireless Communications and Networking Conference (WCNC)*, New Orleans, Louisiana, USA, Mar. 2003.

[4] X. Hong, M. Gerla, G. Pei, and C. Chiang, *A Group Mobility Model for Ad Hoc Wireless Networks*, ACM International Workshop on Modelling and Simulation of Wireless and Mobile Systems, 1999.