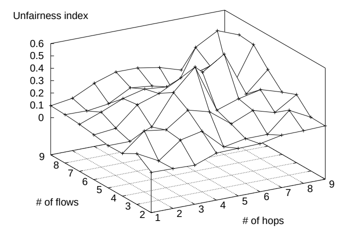
A Path Density Protocol for MANETs

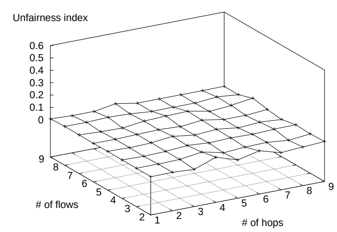
Evgeny Osipov and Christian Tschudin
University of Basel
Computer Science Department
Bernoullistrasse 16, CH–4056 Basel, Switzerland
Email: {evgeny.osipov | christian.tschudin}@unibas.ch

Abstract

Knowing or being able to measure the "path density" at sources of communications is essential to provide fair capacity distribution between sessions in multi-hop ad hoc networks. We propose and have implemented PDP, a path density recording protocol. In this paper we describe PDP, discuss protocol design options and explain how it was piggybacked onto an existing reactive routing scheme. We assess the validity of our approach both using simulations and real world measurements.



A. Plain TCP over IEEE 802.11



B. TCP over IEEE 802.11 with our ingress throttling scheme (see Appendix A)

Figure 1. TCP unfairness index (simulations). See Appendix B for the definition of "unfairness index" and see Figure 5 for the topologies.

1 Introduction

In [1] we presented an adaptive distributed capacity allocation scheme for multi-hop wireless networks. We showed that by throttling the output rate at ingress nodes we achieve both an increase in total network throughput and almost perfect fairness. Figure 1, generated from simulations, shows the degree of improvement with respect to unfairness between multiple TCP sessions in static networks with different maximum number of hops. When our rate bound is implemented we observe that the unfairness virtually vanishes.

In our solution [1] the throttling level is a function of: (a) the number of hops for a particular "connection" (routing path between two peers) and (b) the number of competing connections on the path (*path density*). In the case that a reactive ad-hoc routing protocol is used to provide connectivity, the first parameter of interest is easily obtainable from the route reply (RREP) message received by the source. In this paper we show how the path density parameter can be obtained at run-time during the route establishment phase.

The concept of a path density (PD) implicitly exists in the major quality of service (QoS) architectures developed for the Internet. For instance, in services which require a before hand reservation of network resources, this information can be extracted from the state (either aggregated or per-flow) established by a resource reservation protocol like RSVP. In the simplest case we can count the number of entries identifying the ongoing flows in every router on the path of a particular session and report this number to the source. In MANETs, however, obtaining the path density information is difficult due to frequent topology changes and the specifics of the transmission medium. In MANETs, the competing connections might not share common nodes but still do compete with other connections in the carrier sensing range.

1.1 Contribution and structure of the paper

We developed two variants of a distributed path density protocol: A *stateless* PDP and a *state-full* PDP. To the best of our knowledge, PDP is the only scheme which allows the on-demand gathering of an ad-hoc network state and which was assessed in simulations as well as real world experiments. The design of PDP greatly benefited from being able to instantly switch between simulations and real world experiments during the developing, debugging and performance measuring phases.

The rest of the paper is organized as follows. In Section 2 we introduce the problem of path density gathering and describe major design options for PDP. In Section 2.5 we show how PDP was piggybacked onto an existing routing protocol. After that in Section 3 we report on performance results of PDP obtained using simulations and real-world measurements. We discuss open issues and related work in Section 4 before concluding with Section 5.

2 Measuring the path density

Throughout the paper when talking about a stream of packets between applications at a source and a destination node we will use the terms *connection*, *session* or *flow* interchangeably.

2.1 Problem statement

The problem addressed in this paper is formulated as to discover the number of connections competing for the transmission medium along a path of a particular session. The problem is illustrated in Figure 2. In the figure, Connection 1 is our sample connection which attempts to discover the *path density* along its path. The correct scheme should report five cross-connections plus the main Connection 1 itself. When any two flows partly share their forwarding paths (as is the case with flows one and four in Figure 2) the common nodes should treat them as different connections. However when two or more connections completely share the path from a source to a destination the forwarding nodes should treat this case as one end-to-end session. In the later case the source nodes should perform additional shaping actions as described in [1]. The forwarding nodes of Connection 1 should also take care of distributing the known information about ongoing connections in the corresponding one-hop region. In our example this is needed to introduce the presence of flows two and three to five and six and vice versa, since their forwarding nodes in general might be outside the communication regions of each other.

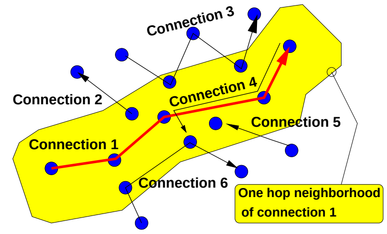


Figure 2. Counting competing connections.

2.2 General solution scheme

We developed two different approaches for gathering the path density information: A *stateless* route request (RREQ) driven and a *state-full* route reply (RREP) driven scheme. Further on we will refer to the first scheme as SL-PDP (StateLess PDP) and to the second scheme as SF-PDP (StateFull PDP). In both approaches every node in the network maintains one state variable *ND* (the neighborhood density). This is the number of cross-connections in the neighborhood of two hops. With every new connection appearing in the neighborhood this value is incremented by one. The state is periodically aged and is decremented by the number of connections which were silent during this period.

The stateless approach does not keep a per-connection state in the forwarding nodes and utilizes the fact that a *route request* message indicates the desire of a node to communicate with another node. In this scheme every RREQ issued by the source advertises the presence of the connection to all nodes which receive its original or rebroadcasted copy. There are two major weaknesses of this scheme that we discovered when experimentally assessing the functional correctness. Firstly, SL-PDP also counts the failed connection setups as existing connections and maintains this information at least for the duration of the aging timer. Secondly, since flooding is used for dissemination of RREQs we cannot control the range of their distribution. As a result, the information about the presence of a connection is spread over more than two hops. We do not discuss further the functionality of SL-PDP in this paper and refer to [2] for more information.

The statefull approach is free from these weaknesses since it counts only active connections and controls the range of information dissemination. The main idea of this scheme is that on reception of a *route reply* message all involved nodes establish a state corresponding to the end-to-end session. A node delays the announcement of a connection to the neighborhood until it sees the first data packet from this connection, as only this event indicates that the connection is active. The announcement of active connections in one hop neighborhood is done every time a node sends or re-broadcasts a route request message. We now describe SF-PDP in more details.

2.3 Statefull path density gathering

In SF-PDP we identify presence of a communication session between two nodes in the network by a source-destination pair (SD). In this scheme every node maintains a table of known SD pairs (*SD_table*). The entries in this table can be of four types: (a) Local active, (b) local inactive, (c) one-hop active and (d) two-hops active as explained in the following subsections. The ND state variable is the number of all "local active" and "non-local active" entries in the SD_table. The major operation of the statefull PDP is shown in Figures 3 and 4.

2.3.1 Adding "local inactive" SD entry

The operation of the statefull PDP during registration of a connection in the network is shown in Figure 3. When a route request message for a particular connection successfully reaches the destination the RREP message is returned to the source. However, reception of the RREP message by a node does not indicate the success of the route establishment procedure since the message itself can be lost on the path to the source. SF-PDP treats the event of RREP reception as an indication of a possible connection through this node. Upon reception of a RREP a SD entry corresponding to this connection is inserted to the SD_table and is marked as inactive.

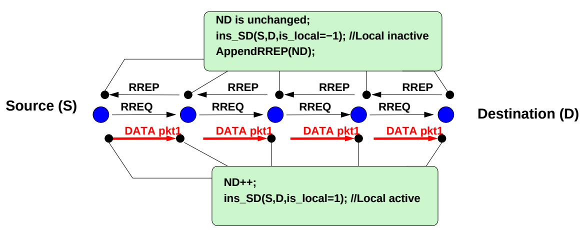


Figure 3. SF-PDP: Processing of RREP and a first packet.

2.3.2 Adding "local active" SD entry

The only definite indication of the active connection in any forwarding node is the event of reception of the first data packet. Therefore, when a local inactive SD record is created, the forwarding engine is instructed to catch the first packet in either direction of the connection.The SD entry becomes active only when the first data packet arrives to the node.

2.3.3 Adding "non-local active" SD entry

The non-local active entries are those which identify connections ongoing in the neighborhood of two hops but not passing through this node. The information about such connections is distributed with broadcast route request messages for any connection originated or re-broadcasted from any of the one-hop neighbors. Figure 4 illustrates the principle of such information dissemination. Before issuing or re-broadcasting the route request message, SF-PDP examines the local SD_table and appends all "local active" and "one-hop active" SD pairs to the RREQ_SD_List. After that it piggybacks the constructed list to the RREQ messages and transmits it.

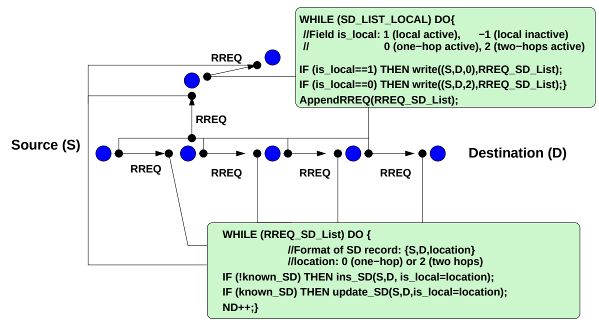


Figure 4. SF-PDP: Processing of RREQs.

When a route request message is received by a node, the SD entries from the RREQ_SD_List are inserted to the local SD_table and marked as "non-local active" of the corresponding location. After that SF-PDP removes the RREQ_SD_List from the received RREQ message and appends its own list as described above. By replacing the RREQ_SD_List with every retransmission we do not allow the information about local connections to spread further than one hop.

2.3.4 Aging of SD entries

Every entry in SD_Table is assigned a timer. The entry is deleted when the timer expires. The duration of the timer is the same as of the route expiration timer.

2.4 End-to-end density reporting and smoothing

The overall goal of PDP is to deliver the path density information to the source nodes. This information is used by the interface queue to configure the delay of emission of locally generated data packets as described in [1].

The path density is the maximum value of ND variables from all forwarding nodes on the path of a connection. The ND value is piggybacked in the route reply message at the destination node and can be modified by every node which receives and forwards the RREP message. When a forwarding node receives a route reply message, PDP examines the local ND state and compares its value to the ND field in the RREP message. The higher value proceeds further to the source.

As we see the path density state is refreshed with every route reply message. Therefore the frequency of its update depends on the frequency of route refreshes initiated by the source. Here we assume that route refreshes are periodically initiated. In the next subsection we discuss how this requirement can be combined with reactive ad-hoc routing protocols.

Due to high dynamics with which new flows might enter and leave the ad-hoc network, the path density reported to the source may fluctuate in time. Therefore, in order to avoid reconfiguring the scheduler on the interface queue too often we allow sources to *smooth* the path density by computing a sliding average.

2.5 Integration in ad-hoc routing protocols

Our primary goal for the implementation of PDP was to avoid introducing additional message exchanges to gather the path density information. This is because from the functional point of view the operations of PDP are very similar to operations of the route establishment phase. Implementing an additional stand-alone protocol for dissemination of PDP information would in the worst case double the capacity already consumed by a routing protocol. We decided to re-use the existing mechanism of message dissemination of the ad-hoc routing schemes.

We integrated PDP in LUNAR [6], which is a L2.5 reactive routing scheme. We also considered possibility of PDP integration in other reactive routing protocols, i.e AODV [4] and DSR [5], and in the proactive OLSR protocol [3]. In this subsection we present our observations on the possibility to use AODV, DSR and OLSR protocols as a PDP carrier.

The route refresh period in proactive routing is normally larger than in reactive schemes. In OLSR, for example, the default refresh period is 16 seconds. We consider it too large assuming high dynamics with which new sessions might enter the ad-hoc network. Based on these observation we do not see proactive routing protocols suitable for dissemination of PDP information.

As for AODV and DSR, we see a number of difficulties for possible integration of PDP in these protocols. As stated in Section 2.1, the flows which share a part of their path to the destination should be treated by our scheme as separate connections. This requirement places a major limitation on the carrier protocol: It should not perform path aggregation. However, this functionality is embedded in both AODV and DSR. Firstly, path aggregation is the normal operation for routes to the same IP subnet. Secondly, even for the cases where IP level aggregation is not used, AODV and DSR provide gratuitous route replies: When a forwarding node is part of the path to a particular destination, it may return the RREP to the source without further propagation of the route request message. Integration of PDP into AODV or DSR would require changes to the core functionality of these L3 routing protocols.

2.6 Integration of PDP in LUNAR

LUNAR is a L2.5 reactive routing scheme. The major difference of LUNAR from AODV and DSR is its position in the TCP/IP protocol stack. Since LUNAR is located below IP layer, the IP level route aggregation which is present in L3 routing schemes is impossible. The route request messages in LUNAR always propagate from the source to the destination. That is, gratuitous replies are not used either. LUNAR is well suited for integration with PDP also because of its high dynamics of information update: LUNAR re-establishes the entire path every three seconds.

3 Experiments

The single code base both for the Linux kernel and the network simulator ns-2 [12] allowed us to debug and extensively test the PDP+LUNAR functionality on a wide range of static and dynamic simulation scenarios. We implemented and tested both stateless and statefull PDP schemes. Once we obtained a stable protocol we generated a real world version and performed a correctness test in a test-bed as described in Appendix C. While performing a preliminary evaluation of SL-PDP in the simulator we found the drawbacks of this scheme described in Section 2.2. As a result we decided to go on with SF-PDP. In this section we present the results from testing the protocol both in simulations and the real-world test-bed.

3.1 Metrics

In all experiments we used correctness and convergence time metrics to evaluate the performance of PDP. Correctness is evaluated with respect to the "path density" as reported to the end nodes of the session (the path density in our experiments was also reported to destinations because we used bidirectional traffic in the experiments).

3.2 The effect of path density (Simulation)

First, we show the effect of path density on the unfairness index and total TCP throughput in networks with different number of active TCP flows. We performed a series of simulations in ns-2 (see Appendix C for setup details) on a set of three hop networks depicted in Figure 5. We varied the number of connections in the network from 2 to 9; For each case we run 30 simulations with enabled and disabled reaction on the reported path density. The dynamics of the unfairness index (see (2) in Appendix B) in both cases is shown in the part of surfaces corresponding to three hops and 2 – 9 connections in Figure 1.

Under our scheme the fairness among competing TCP flows was close to perfect, while ignoring the path density we observed an unfairness of up to 25%. Note that the nature of the used unfairness index reflects the closeness of individual goodputs of each flow to the ideal share, therefore the closer the unfairness index to zero the higher is the minimal individual goodput.

Number of connections	Total TCP throughput, kb/s	
	Without path density	With path density
2	419	419
3	405	407
4	396	407
5	395	406
6	394	406
7	386	407
8	375	406
9	380	406

Table 1. The effect of path density on total TCP throughput.

Table 1 shows the effect of ingress rate throttling on total TCP throughput depending on the path density for the two cases. As we observe from the table the total TCP throughput in the network is higher when accounting for the path density. This implies better network utilization when all competing flows fully utilize their allocated share. Note that in reality we will also have a number of short-living communications and flows with a number of short transmission bursts e.g. web browsing. In this case the share of capacity computed based on the path density reported by PDP will not be fully utilized. When several connections are originated from the same source the leftover capacity can be reused by the active flows, otherwise the network will be underutilized until the route for the inactive connection will be removed. Therefore it is essential to dynamically update the ingress nodes with the path density information. In the case of our PDP+LUNAR combination the information about inactive flows will be aged within three seconds in all PDP-aware forwarding nodes.

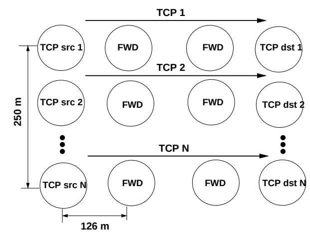


Figure 5. A set of three hop networks for illustration of the path density effect.

3.3 Static Scenarios (Real World)

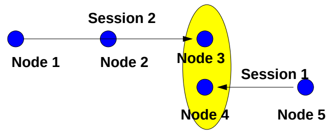
We present an evaluation of the functional correctness of PDP on two static scenarios depicted in Figure 6. While all nodes in reality are located in the reception range of each other (hence the same radio interference domain) we configured LUNAR so that a node can hear the data transmission only from a selected set of nodes and discard packets from others. In this sense only Nodes 3 and 4 in Figure 6*a* "share" the regions of assured data reception of each other. In the second case (Figure 6*b*) Node 2 is able to hear transmissions from Node 4 and Node 3 from Node 5. PDP should, however, report the same density information on all nodes.

In both scenarios Session 1 started at time 1 s and finishes at time 30 seconds. Session 2 establishes the path 10 seconds after Session 1. The duration of Session 2 is 10 seconds. We used the ping protocol to generate traffic of the corresponding sessions. The interval between two consequent ICMP requests is 100 milliseconds. We repeated both real world experiments up to 10 times and obtained the same behavior of our two metrics. Figure 7 shows the results from one of these runs.

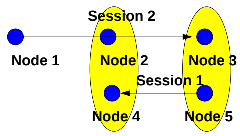
As is visible from the time diagrams, PDP correctly reports the path density to the corresponding end nodes of the two connections within 1.5 – 3 seconds from the start time of a session. The delay in dissemination of the information is explained by the default route refresh interval in LUNAR. After the session is established the earliest time a new RREQ message is generated by the destination in the backwards direction is after 1.5 seconds.

3.4 Dynamic Scenario (Simulation)

We evaluated the behavior of PDP in presence of mobility in simulations only. This time we seek a quantitative assessment of the path density information provided by PDP. We studied the scenario shown in Figure 8.

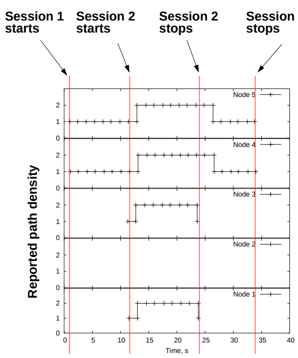


a) Topology 1.

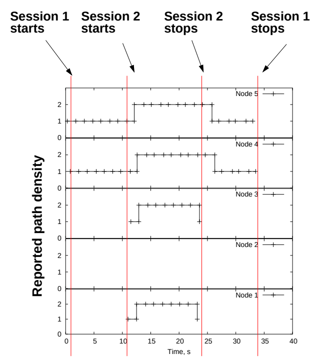


b) Topology 2.

Figure 6. Topologies for real world-experiments.



a) For scenario in Figure 6a.



b) For scenario in Figure 6b.

Figure 7. Reported path density in real-world experiments.

In the scenario we have three TCP sessions each following a path of two hops. Initially all nodes are located outside the range of assured reception of each other. After five seconds from the simulation start the middle nodes of sessions one and three begin a movement towards the middle node of Session 2. The speed of the nodes is 10 m/s. In their final destination both Nodes 2 and 8 are in the range of assured reception of each other and Node 5. The mobile nodes remain in this position for 80 seconds; After that they start moving back to their original position.

The goal of this experiment is to evaluate the dynamics of PDP based source throttling. We performed two experiments on this scenario. First, the path density information was ignored and TCP flows might transmit without rate limitation. In the second experiment we configured the scheduler on the interface queue with a throttling bound dynamically computed according to the path density information reported by PDP. Figure 9 presents the results of these experiments.

As we observe from the figure, the slope of TCP sequence numbers curves is the same for all three flows provided path density is taken into account, which results in fair sharing of the network's capacity. Moreover, in this case the progress of all TCP flows is smooth and free from interruptions when compared to the case where the path density information is ignored. In addition to the smoothness of the TCP flows, the resulting total TCP throughput (not shown in the figure) remains the same as in the case without rate limitation at sources.

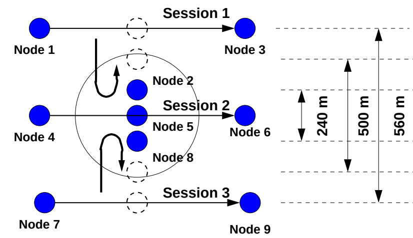


Figure 8. Topology 3 for experiments on a dynamic scenario.

To sum up, the results of the last experiments illustrate our overall goal: If flows in a MANET are aware about the presence of each other and reduce their rate accordingly, none of the competitors is able to capture the capacity as it happens for the plain combination of MAC 802.11 and TCP. The latter case is represented in Figure 9 by flow TCP 2 (w/o path density) which worsens the performance of TCP flows 3 and especially 1.

4 Discussion and related work

Our path density gathering scheme presented in this paper is a distributed protocol for the on-demand discovery of an ad-hoc network state. The PDP protocol plus our ingress rate throttling approach permits to implement a capacity allocation scheme with guarantees on fairness of communications.

In addition to the admission control in sources we see potentials of our protocol for congestion-aware routing. Indeed, the ND state kept in all forwarding nodes is an excellent indicator of the neighborhood's load. If a forwarding node – instead of re-broadcasting the newly arriving route request – would first examine the neighborhood density, it can estimate the chance for this flow to obtain an acceptable service while being forwarded through this area. If a neighborhood is overloaded with existing connections the route request can simply be discarded. Assuming a network with relatively high node density, the "surviving" route requests would discover less loaded paths.

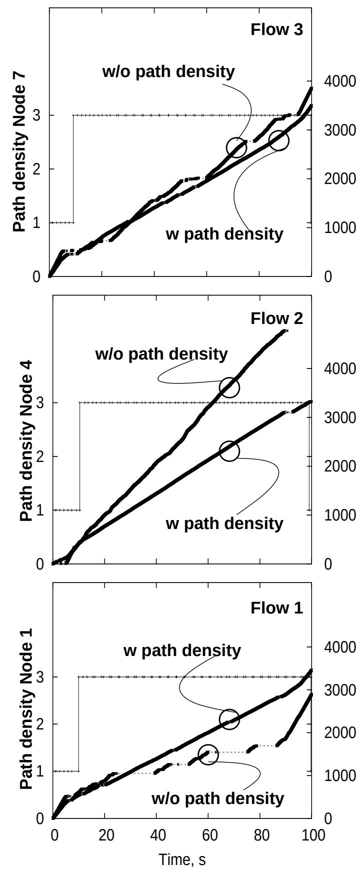


Figure 9. Path density and TCP sequence numbers progress for scenario in Figure 8.

4.1 Related work

The idea of estimating the state of an ad-hoc network is not new. There are several approaches based on distributed network state exchange. In [7] SWAN, a distributed QoS architecture for MANETs is proposed. The proposed admission control at sources as well as the rate limitation scheme in all nodes is based on hop-by-hop measurements of the available bandwidth. The authors suggest to use a reactive routing protocol to gather the results of these measurements along a path of a connection. New flows are admitted for transmission only if the reported path capacity is less than a certain threshold.

In [8] the authors suggest to exchange the occupancy level of the interface queue of every node in one and two hops neighborhood. This information is then used to construct a virtual queue of the neighborhood and to coordinate dropping of packets. The exact mechanism to exchange such information is however not described in the paper. The authors suggest to use additional message exchanges for the dissemination of the information.

In [9] the authors discuss a distributed weighted fair queuing algorithm, similar to its analog in the fixed Internet. In order to distribute the weights to the interface queue of each node in the neighborhood of one hop, the authors suggest to encode this information in the MAC 802.11 header.

5 Conclusions

In this paper we considered the problem of measuring the “path density” in MANETs. We presented an approach for gathering such information in a distributed way at run-time. We understood that building PDP based on observing route request transmissions only leads to incorrect counting of the failed connections and resorted to storing per-connection state in the forwarding nodes. We showed how PDP can be piggybacked inside LUNAR messages without introducing new transmission events and tested our implementation with combined simulation and real-world measurements.

Acknowledgments

We wish to thank an anonymous reviewer for the detailed comments which greatly helped the preparation of the camera-ready version of this paper.

References

[1] E. Osipov, C. Jelger, Ch.Tschudin, “TCP capture avoidance in wireless networks based on path length and path density”, Technical Report CS-2005-003, University of Basel, Switzerland, April 2005.

[2] E. Osipov and Ch. Tschudin “A path density protocol for MANETs”, Technical Report CS-2005-004, University of Basel, Switzerland, May 2005.

[3] T. Clausen, P. Jacquet, “Optimized link state routing protocol (OLSR)”, RFC 3626, IETF, Oct 2003.

[4] C. Perkins, E. Belding-Royer and S. Das, “Ad hoc on-demand distance vector (AODV) routing”, RFC 3561, IETF, Jul 2003.

[5] D.B.Johnson, D.A.Maltz,Y-C.Hu, “The dynamic source routing protocol for mobile ad hoc networks (DSR)”, IETF draft (work in progress), 2003. [Online]. Available: `http://www-2.cs.cmu.edu/~dmaltz/dsr.html`.

[6] Ch. Tschudin, R. Gold, O. Rensfelt and O. Wibling, "LUNAR: a lightweight underlay network ad-hoc routing protocol and implementation", In *Proc. NEW2AN'04*, St. Petersburg, Russia, Feb. 2004.

[7] G-S. Ahn, A. T. Campbell, A. Veres, L. Sun, "SWAN", IETF draft (work in progress), 2003. [Online]. Available: `http://comet.ctr.columbia.edu/swan/draft-ahn-swan-manet-00.txt`.

[8] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED" , In *Proc. MobiHoc'03*, Annapolis, MD, USA, 2003.

[9] N.H. Vaidya, P. Bahl and S. Gupta, "Distributed fair scheduling in a wireless LAN", In *Proc. MobiCom'00*,Boston, MA, USA 1999.

[10] K. Xu, S. Bae, S. Lee and M. Gerla, "TCP behavior across multi-hop wireless networks and the wired Internet" , In *Proc. WoWMoM'02*, Atlanta, GA, USA, Sep. 2002.

[11] Uppsala University Ad Hoc Implementation Portal, [Online]. Available: `http://core.it.uu.se/AdHoc/ImplementationPortal`.

[12] Network Simulator NS-2, [Online]. Available: `http://www.isi.edu/nsnam/ns/`.

Appendix A - Adaptive capacity allocation scheme for capture-free communications

TCP capture, as is described in [10], is one of the unsolved problems in multi-hop ad hoc networks which results in extremely unfair distribution of network bandwidth between competing sessions.

In [1] we suggested a scheme for dynamic capacity distribution between competing connections that partly or completely share communication regions. We proposed the ingress throttling mechanism which guarantees capture-free communications for all involved flows.

The input information to our scheme is the *number of competing connections* on the path of a particular multi-hop TCP connection (*path density*) and the knowledge of the saturation point of the radio transmission medium beyond which a single TCP flow starts to loose packets (*critical network load*). This information is used to compute the output rate at ingress nodes (1).

$$r_{ingress} = \frac{\alpha \cdot L^{crit}_{TCP}}{h \cdot N}. \quad (1)$$

In (1) L^{crit}_{TCP} is the estimated rate of TCP data segments which creates the critical network load, h is the number of hops of a particular flow and N is the path density. If all sources follow this rule, then the total network capacity is fairly distributed between the competing flows as shown in Figure 1.

Appendix B - TCP unfairness index used in Figure 1

We define the unfairness index u used in Figure 1 as the normalized distance (2) of the actual throughput of each flow from the corresponding optimal value.

$$u = \frac{\sqrt{\sum_{i=1..n} (X_{opt_i} - X_{act_i})^2}}{\sqrt{\sum X^2_{opt_i}}}. \quad (2)$$

In this formula X_{opt_i} is the ideal throughput of flow i obtained under fair share of the network capacity. In order to compute this value we divide the throughput of the corresponding flow obtained when running alone in the network by the number of competing flows. X_{act_i} is the actual throughput of the same flow achieved while competing with other flows. This index takes the values between 0 and 1 and reflects the degree of global user dissatisfaction, hence the value of 0 corresponds to perfectly fair communications and 1 represents the opposite case. Note that the index is useful only when applied to two or more flows. This is also reflected in Figure 1, where the y-axis starts from two connections.

Appendix C - Experiment Setups

The real-world results were obtained from a setup of five DELL Latitude laptops with ZyXEL ZyAIR B-100 wireless interfaces. We use the Linux operating system with 2.6 kernel and LUNAR implementation available from [11].

The simulation results were obtained using *ns* simulator of version 2.27. In simulations with scenarios depicted in Figures 5 and 8 we used TCP Newreno as the most popular variant of the protocol. We set the value of the TCP maximum segment size (MSS) to 600 B. The data transmission rate of all devices is 2 Mb/s. Other ns-2.27 parameters have default values.