CrossTalk: A Data Dissemination-based Cross-layer Architecture for Mobile Ad-hoc Networks

Rolf Winter, Jochen Schiller
Institute of Computer Science
Freie Universität Berlin, Germany
{winter, schiller}@inf.fu-berlin.de

Navid Nikaein, Christian Bonnet
Mobile Communications Department
Institut Eurécom, France
{nikaeinn,bonnet}@eurecom.fr

Abstract - **Unlike infrastructure-based networks, mobile ad-hoc networks suffer from severe performance problems due to their dynamic nature. The medium is shared and interference-prone, routes are unstable, energy can be a limiting factor for devices such as sensor nodes just to name one. To overcome those problems, cross-layer architectures are a promising new approach. They can reduce the effect of the aforementioned problems and therefore increase scalability and reliability of such networks. This paper introduces CrossTalk, a cross-layer architecture based on data dissemination that enables each node in an ad-hoc network to evaluate its own status locally against that of the network for decision processes such as routing. Furthermore, we propose a load balancing algorithm based on our architecture that achieves significant improvements in load distribution and packet delivery delay.**

I. Introduction

Layered approaches as used in most modern networking environments have certain characteristics which make them currently the primary architectural choice when designing a new protocol stack. Their key advantages are the low design complexity, the modularity and an improved maintainability compared to monolithic stacks. Since each layer has a well defined functionality, designing each layer can be done without worrying about specific functionality of upper or lower layers. This modularity allows for the combination of different protocols, thereby helping to construct network stacks tailored towards different networking environments. Layered protocol stacks are easier to maintain since errors can be traced back faster to a certain layer, which in turn can be updated, modified or exchanged more easily. These characteristics are predominant in commercially operated, large, infrastructure-based, centrally administered and reliable networks and are the reason for the success and longevity of layered approaches. On the other hand, strongly layered approaches leave out various possibilities to improve the performance of the network stack and do not support applications with lower layer information and vice versa.

In ad-hoc networks performance and scalability are key issues. They are affected by many factors intrinsic to ad-hoc networks, such as the shared, unreliable medium resulting in bit errors, collisions, high delays and lowered throughput. In addition, the fact that devices in such networks are likely to be battery-driven and relatively weak in terms of computational power imposes special constraints on the protocol stack. The mobility of nodes also plays a significant role. It affects the stability of routes through the network, possibly resulting in broadcast storms which consume large amounts of the available and scarce bandwidth. In such dynamic environments, cross-layer approaches are promising since the possible performance gains can significantly improve the scalability, the delay performance and the throughput.

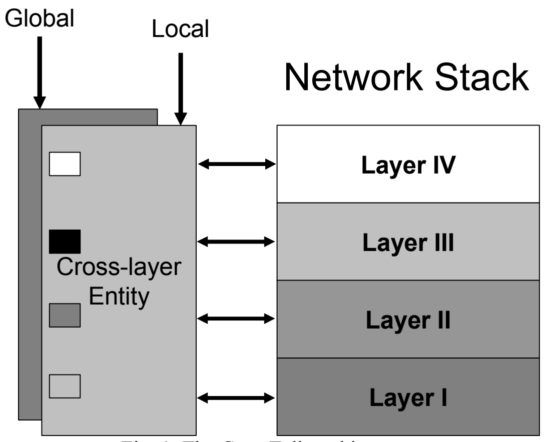

Fig. 1. The CrossTalk architecture

Cross-layering is not the simple replacement of a layered architecture nor is it the simple combination of layered functionality. Cross-layering tries to share information amongst different layers, which can be used as input for algorithms, for decision processes, for computations, and adaptations. This process of sharing has to be coordinated and structured somehow since cross-layering could potentially worsen the performance problem that it intends to solve. This is due to several effects. Optimization processes at different layers could try to optimize a common metric in opposite directions. Furthermore, two different metrics could have negative impacts on each other when trying to optimize them, such as energy efficiency and delay. A general problem is that altering a metric at one layer often has an effect on other layers implicitly. For example, altering the transmission power on the physical layer can have an effect on the network layer as nodes might disappear from the direct transmission range.

So instead of a replacement, cross-layering is the enhancement of the traditional layered architecture. With cross-layering every layer and system component can access and provide information in a structured way, controlled by a coordinating entity spanning the entire layered stack (compare Fig. 1). This way the advantages of the layered approach can be largely preserved, guaranteeing the longevity of the architecture.

In addition to the performance improvements at lower layers, cross-layering allows us to design new kinds of applications. Especially affected are distributed applications and applications sensitive to changing network conditions such as QoS-sensitive multimedia applications. The information provided by the cross-layer architecture could support decisions about where to place objects and services or the choice of algorithms such as algorithms for compression or error correction.

The novel aspect of our cross-layer design is that it tries to establish a network-wide, global view of one or multiple metrics like load, battery status or degree in a distributed fashion on every node in the network. Having such a global view of the network allows a node to evaluate its own status against the average status within the network at any time. For example, a node could conclude whether it carries more load than the average node and how much it is overloaded compared to the average. Having this information, it can then use it for decision processes such as routing, load balancing, position estimation and so forth.

In ad-hoc networks a lot of wasteful operations in terms of resources such as bandwidth have to be carried out. That includes for example routing processes, where often, using broadcast mechanisms, the whole network is involved to find a route but only a few nodes take part in the actual routing process for data packets after the route is established. On the other hand, local operations are lightweight, but they lack accuracy and ultimately can be inefficient. For example, if a node is able to increase its performance locally by some means (e.g. by boosting its output power), it might at the same time significantly increase the interference with its neighbors. On a multi-hop path, that might effectively lead to a lower overall performance. Therefore the basis of our work is one general principle: Act locally considering the global network status [1]. This way simple local actions achieve global objectives [2]. We show the effectiveness of this principle by applying the global view approach to a load balancing scheme which also addresses the problem of rebroadcast redundancy.

The remainder of this paper is organized as follows. Section II discusses related work. In Section III, we present the details of the CrossTalk architecture together with the load balancing reference application. Section IV deals with the verification of the architecture and the application of it by analyzing and evaluating experimental results. Section V concludes this paper and gives a brief outlook on our future work.

II. Related Work

Cross-layer design is becoming an increasingly investigated research area. Various aspects within this field have been studied. The research carried out so far reflects the diversity of the problems caused by the system dynamics in ad-hoc networks. Therefore, we do not present an exhaustive overview here but merely the application domains of cross-layering and related work for our reference application.

Ad-hoc networks are only one suitable application domain for cross-layer design. In general, the most appropriate class of networks for cross-layering are wireless networks, including cellular [3] and sensor networks [4].

The specific problems analyzed comprise power and topology control protocols [5], energy efficiency [4] and Quality of Service (QoS) [6][7][8], to name a few. The work spans from architectural thoughts [9][10] to detailed simulations of performance gains and novel applications [11]. However, none of the existing approaches try to establish a global view of the network to meet the challenges of ad-hoc networks.

At the same time that cross-layering is increasingly introduced in systems design, it has received some legitimate criticism for its intrinsic potential problems [12]. That includes unstructured design and code, unintended effects on other system components and layers, adaptation loops and stability issues.

Load balancing and the reduction of the problems caused by the broadcast storm problem remain an investigated research area. Zhou et al. designed [14] a cross-layer framework together with a routing protocol to reduce redundant broadcasts. Their algorithm PRDS-MR is dependant on some positioning system, which our scheme is independent of.

In [15] load balancing is done implicitly by avoiding the core of the network. Here, as well, some position information has to be known including certain knowledge about the topology since the distance of a node to the core of the network is utilized as a metric. DLAR [16] works in a similar way to other protocols that try to optimize routes according to a certain metric. Every node adds its local value of that metric to the route request. The destination node then has to wait for a certain time before choosing the most appropriate path, in the case of DLAR the least loaded route. In that respect LBAR [17] works in a similar way. The advantage of our approach is that the route selection phase at the destination is unnecessary with our approach since we use the global view to act already during the route discovery process.

III. System Design

As already mentioned, the general, novel idea of the CrossTalk architecture is to establish a network-wide, global view of a metric such as load at every node in the network. In addition, the information from layers and system components are made available locally to be utilized as for example proposed in [14].

A. The CrossTalk architecture

The CrossTalk architecture consists of two views (see Fig. 1). There is the local view containing node specific information. This is information contributed by each layer of the stack or system component and can be used for local optimizations as for example in [5] or novel applications [11]. This information could include current battery status, signal to noise ratio (SNR), bit error rate, one hop neighbor count just to name a few.

Additionally, there is the global view that is constructed from information gathered by our data dissemination process. First of all, the local information has to be propagated. To keep the overhead low, no extra messages are sent. Instead the local information taken from the local view is piggybacked onto outgoing packets, keeping the overhead at a minimum. Only the source of a packet is adding its local information. Forwarding nodes do not include their information on top. By doing this, there is only a slight increase in the packet size, resulting in a small overall footprint of our system. Every node inspects received packets for that information, extracts it and adds it to its global view. This way, the global view collects numerous samples of local information from various nodes within the network. The samples can be augmented with additional information to give them a weight such as distance if that information is available. Also, they are given a timestamp since samples are purged after their useful lifetime. Clearly, following this approach the global view will never be 100% correct and that is not what we aim at. Instead we want to have a reasonably up-to-date and correct view of the network, which allows a node to evaluate its own status. For example, if a node runs at 50% of its capacity, that number as such has no immediate meaning. However, if the rest of the nodes in the network run at 10%, that means that the node is clearly overloaded. Therefore, with our approach we want to provide a node with information to compare its own local information with in order to be able to act upon that comparison.

Part of the global view are algorithms to compute the network-wide view from the collected samples. We evaluated two general types of algorithms one being the simple mean value and the second type being weighted moving averages. For both types we considered different variants, each one following a different intuition. For one variant we excluded local information samples from one-hop neighbors. The intuition behind this is that direct neighbors can have a similar value for local information. For example, the degree (number of neighbors) could be similar due to the proximity of the nodes, eventually influencing the computed mean value. The weighted moving average variants vary in weighting factor and weighting function:

$$v_{global} = \frac{\sum_{i=1}^{i=n} w_i s_i}{\sum_{i=1}^{i=n} w_i}$$

The above formula denotes the general structure of a weighted moving average, where *n* is the number of samples, *w* is the weight and *s* is the sample value. *w* is calculated using a linear, triangular or exponential function with distance or time as a metric (see Fig. 2 for an example). The intuition behind the linear distance weighted function is that the further away a node is from the node trying to establish the global view, the less information the node will get from that node and the more the value might differ due to the distance. Therefore, such a sample will get more weight than a sample from a node close by. The triangular distance weighted function has a slightly different intuition. The values furthest away will most likely come from nodes at the edge of the network. Here, due to lower node density or other effects, the values might not reflect the actual average well and are therefore weighted less. Nodes close by are also weighted less for the same reasons that neighbors are excluded, that is, that the proximity of the nodes might be reflected in the disseminated information. The other weighting factor is time. The more recent a sample is, the higher it should be weighted since it more accurately reflects the current state of the network. With time as a weighting factor, we evaluated two weighting functions. One is a linear function giving the highest weight to the most recent sample and the other one gives exponentially more weight to more recent samples.

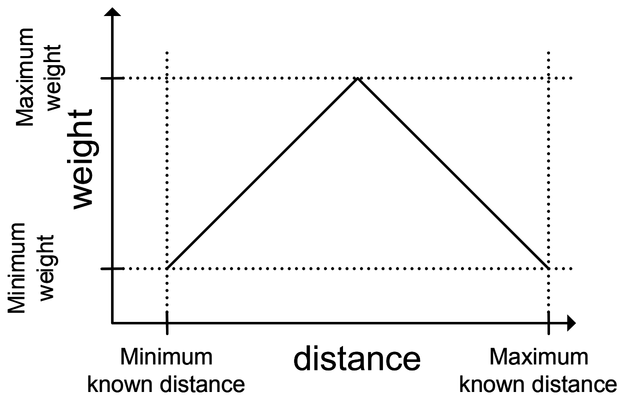

Fig. 2. Triangular weighting function

As summary, our architecture consists of two views both responsible to hold, organize and provide data. In addition, they provide a set of algorithms to compute more significant data, which in turn can be accessed and utilized as shown in the following section.

B. Reference application

In this section we exemplarily apply our architecture to a load balancing and load reduction algorithm using AODV for routing. Balancing the load in ad-hoc networks is important since nodes with a high load burden deplete their batteries quickly, thereby increasing the probability of disconnecting or partitioning the network. Additionally, if the network is more balanced, the spatial reuse of the spectrum allows for a higher throughput and relieves the center of the network [18]. The second aspect of our algorithm, the load reduction, tries to attenuate the effects of the broadcast storm problem [13], which are a higher collision probability of packets and unnecessary resource consumption.

To support the decisions our algorithm has to make, we distribute the local load using our data dissemination process. To calculate the local load we count the number of packets sent by a node on behalf of other nodes during a fixed time frame or slot *t* (see Fig. 3). We do not include our own packets to provide a degree of fairness. To eliminate fluctuations which might have occurred during a slot, we use *n* slots to calculate the actual load over a time period of *n* * *t*. To account for changes in the load during that period, we calculate the weighted moving average of all slots except the current slot with a weight being equal to the slot number as shown in Fig. 3.

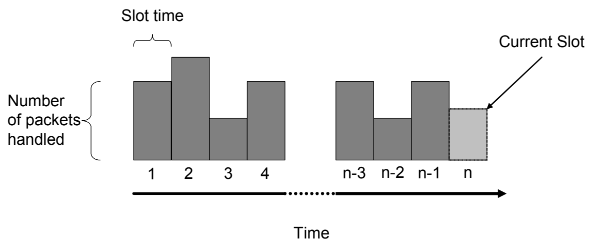

Fig. 3. Load calculation

The algorithm itself is a two-phase algorithm and works as follows:

Whenever an AODV route request reaches a node, it calculates the global view and compares its own local load against that view. If the node is not overloaded compared to the global view, it resumes "normal" operation in terms of AODV. When the node finds itself overloaded, it calculates the *overload degree*, that is the ratio of the own local load and the global view of the load. From the point of being overloaded (overload degree > 1) up to a predefined threshold, the *delay bound* (compare Fig. 4), the node will hold back the route request for a certain amount of time before forwarding it. This delay grows proportionally with the overload degree up to the delay bound, where the delay reaches its maximum. By delaying the route request, the probability increases that an alternative route (through other nodes) will be established circumventing the overloaded area, which is likely to be the core of the network [18]. This way the following data packets will not have to be forwarded by the overloaded node, which would have even further increased the load and at the same time increased the collision probability with other packets in the overloaded area. With the delay being proportional to the overload degree, a route will be established balanced between being short in terms of the hop count and carrying mild load. Once its delay bound is reached, a node holds back packets with a maximum delay *d*. Even if the load is beyond the delay bound, packets will not be delayed longer than *d*.

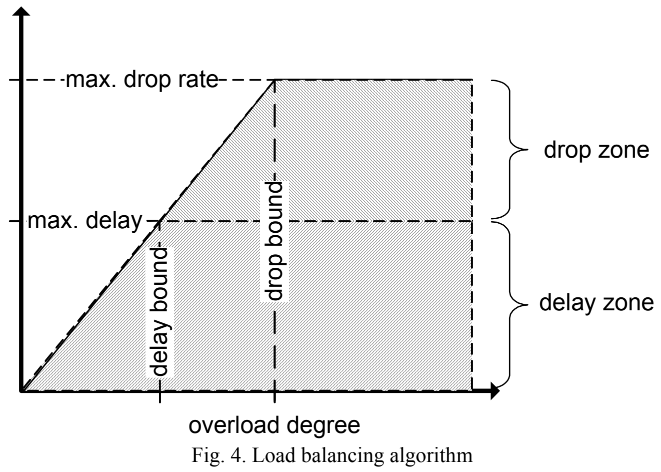

Fig. 4. Load balancing algorithm

The second phase of our algorithm covers the load reduction functionality. When a node has an overload degree beyond the delay bound, it starts dropping route requests at a certain rate, otherwise delaying them for the maximum delay time *d*. The drop rate also grows proportionally with the overload degree up to a certain threshold, the *drop bound*. Beyond the drop bound, route requests will be dropped at a predefined maximum drop rate *r* or delayed by *d*. By dropping route requests, the heavily overloaded nodes make sure that the path towards the requested destination will not lead through highly overloaded nodes. Additionally, the broadcast storm problem in highly overloaded zones is reduced. The maximum drop rate should never reach 100% since an extremely overloaded node might be so for the simple reason that it is the only node connecting parts of a network. Since it does not accept every route request, the node keeps up a certain path quality for existing paths through it, by limiting the amount of paths through that node.

IV. Experimental Results

The first paragraph of this section deals solely with the experimental validation of the global view. Before it can be applied, it has to be proven to be correct and functional within certain bounds. The second part of this section analyzes the experimental results of our load balancing and load reduction algorithm. The analysis itself is based on simulations using ns-2 together with AODV-UU [19]. For every data point in a graph, 20 simulation runs were averaged to compensate for marginal phenomena. Every run in turn was running for 600 simulated seconds.

For the experimental evaluation, we wanted to isolate the effects of various parameters such as network size, network density, mobility and more on the global view and our reference application. For this reason, we keep all parameters fixed except the one we want to evaluate. The standard simulation network is a static, 200 node network on a 2000x2000 meter square plain. Each node has a transmission range of 250 meters. The nodes are placed randomly onto the plain, each running a traffic pattern agent that searches for a new destination every 10 seconds and sends one packet per second afterwards. For the reference application, the delay bound was chosen to be 1.8 and the drop bound 2.5 using a maximum delay of 100ms and a maximum drop rate of 67%.

A. Global view

For our experimental evaluation, we added thresholds for the calculation of the global view. Whenever the global view contains an insufficient number of samples, our cross-layer entity will not calculate a network-wide average on request forcing "normal" mode. The composition of the samples is also evaluated. If more than two-thirds of the samples are from direct physical neighbors, the cross-layer entity will not calculate a global view for that metric preventing cross-layer operation, either.

When a node joins the network, we added an initialization procedure. The joining node would broadcast an initialization request to its neighbors. The neighbors reply with a message

containing their global view, which in turn is used at the newly joined node to fill the first slot of the load calculation algorithm.
We also exploit the fact that AODV is sending periodic hello messages. Instead of only using this kind of message to distribute our own local information, we also enrich hello messages alternatingly with information of our one hop neighbors. This way, we use the one-hop broadcast hello to disseminate recent and close by information more effectively.

We identified several parameters for our experiments. Since the global view's correctness is based on the amount and quality of the samples, the network structure has to be considered. For this purpose, we evaluated the parameters density (50 – 150 nodes/km²), network size (50 – 400 nodes), mobility (1.4m/s) and topology geometry (1:1 – 1:9), which in our experiments is the ratio of the topology's width and height. Since we do not generate any messages, the global view is also dependant on the traffic pattern and on the load. Therefore, we tested our CrossTalk architecture under different load scenarios (0.5 – 2 pkts/s). The traffic pattern we apply distinguishes between local and distant communication. Within a certain radius (in hops), a node considers the communication local, whereas beyond that radius the traffic is considered distant traffic. In our simulations, we evaluate different ratios of local and distant traffic (25% - 100% within 3 hops). Finally, we analyzed the global view under churn conditions for the simple reason that the global view will become more accurate over time since it most probably aggregated more samples (each node fails every 60 – 250s). With churn, there are constantly nodes joining the network with an empty global view influencing the correctness of the local evaluation process.

We show the quality of the global view using three different metrics which have been analyzed for all the scenarios described above. Due to space limitation and the similarity of the results we are not showing all results graphically here. The first metric to show the global view's accuracy is the average value of all global views. This value is compared with the average value of all local views in the network. Ideally the two values should match.

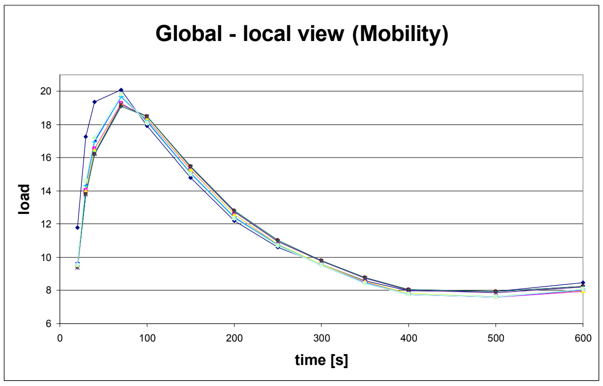

Fig. 5. Global and local view comparison

The average alone does not reflect the quality of the global view accurately enough. The second metric reflects the global view's uniformity across nodes, which is the standard deviation of the global view at each node in the network. This value is compared against the standard deviation of the local view at each node in the network. Ideally, the global views' standard deviation is zero, independently from the local views' standard deviation.

The third metric we termed correctness. We calculate the average local view artificially and then compare the local view of each individual node against it. If this comparison and the comparison of local and global view at each node yield the same result (overloaded vs. not overloaded), the node evaluates its status correctly, otherwise it fails to do so. The correctness is the percentage of nodes in the network that evaluate their status correctly.

Fig. 5 shows the impact on the global view over a 600 seconds simulation run with mobility. The chosen mobility model was the random waypoint model with a minimum and maximum speed of 1.4m/s, which is fast walking speed, and 20 seconds pause time between two node movements. We are aware that random mobility is not a realistic model. However, it is a worst case mobility model for our reference application analyzed later on. For reasons of clarity, we omitted the legend since the curves shown in the graph are extremely close. The figure shows all analyzed global view algorithms together with the actual average of all local views within the network. The closer the graphs are together, the more accurate the global view is. As can be seen, the graphs are very close showing the precision of all global view algorithms. In all other tested scenarios the global view performs similarly and therefore the graphs are omitted, as mentioned before.

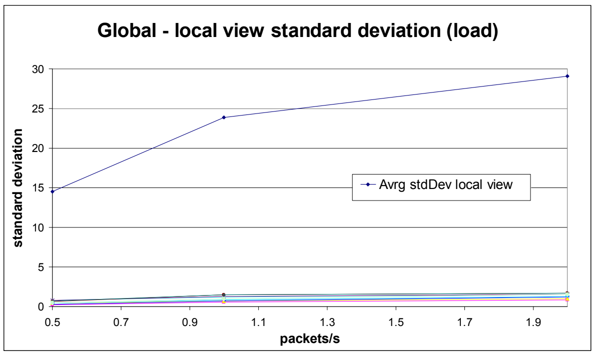

Fig. 6. Global and local view' standard deviation

Fig. 6 depicts the dependency of the standard deviation of the global view algorithms. As can be clearly seen, the algorithms are robust against load differences. The results shown are similar in all the tested scenarios tested. If we zoom into the graph, we would see that the best performing algorithm is the simple mean value followed by the time weighted averages. In relative dimensions the simple mean algorithm outperforms the distance weighted averages by a factor of 3. Absolute, these differences are marginal though.

The correctness of the global view is pivotal. As stated before, we do not aim at having a 100% accurate view of the network but a correctness well above 90% is our goal. Fig. 7 displays the correctness in differently sized networks and under churn over a 600s simulation run. We omitted the variants that leave out samples from neighbors since they were similarly performing to their corresponding counterparts. Here again the simple mean value outperforms the other algorithms. The main advantage though can be seen in small networks. With increasing network size, the distance based algorithms gain. The graph from the churn scenario shows that in the beginning of the simulations, when all the nodes are initializing their global view, the time based algorithms perform best since they more accurately reflect the sudden load increase in the network (up to 8 % more accurate).

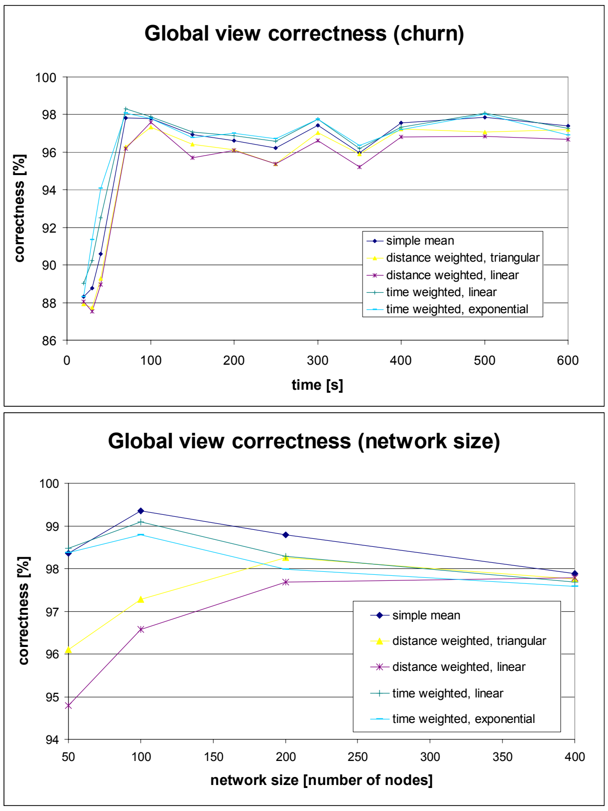

Fig. 7. Global views' correctness

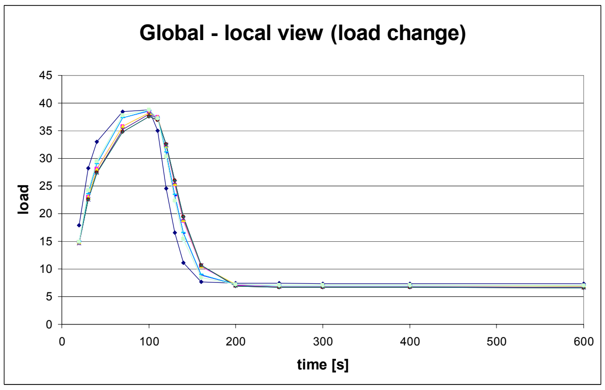

Fig. 8. Global and local view under load change

Finally we evaluated how reactive the global view algorithms are to extreme changes of the observed metric. The beginning of a simulation can be seen as an extreme increase in traffic (all nodes starting at zero) and from the simulations we could analyze the reactivity to that (see Fig. 6). We then evaluated the reactivity to extreme drops in traffic. We let applications run at each node for the first 100 seconds of a simulation run and then reduced the packet rate by a factor of 5 and the lookup rate for new destinations by 2. The results are displayed in Fig. 8, clearly showing the ability to react to extreme metric changes.

B. Load balancing

Our load balancing algorithm was also tested in all the scenarios described in the previous section. An important thing we wanted to make sure was that our cross-layer approach does not perform worse than the "traditional" approach. Performance was measured according to the following metrics. The average number of messages sent per node during a simulation run reflects whether there is an actual load reduction effect or not. The on average most overloaded node reflects the reduction of the bottlenecks within the network. The coefficient of variance, which is the ratio of the standard deviation to the mean multiplied by 100, shows the actual load balancing effect amongst nodes when applied to the average number of packets send per node. Additionally, we measured the average delay per hop, which is influenced by both effects (load reduction and load balancing) since both reduce the likelihood of collisions. Since we drop packets, there is the potential problem that our packet delivery ratio (PDR) declines. Therefore, we also monitored the PDR. To calculate the global view we followed the KISS (Keep It Simple, Stupid) principle and applied the simple mean value algorithm.

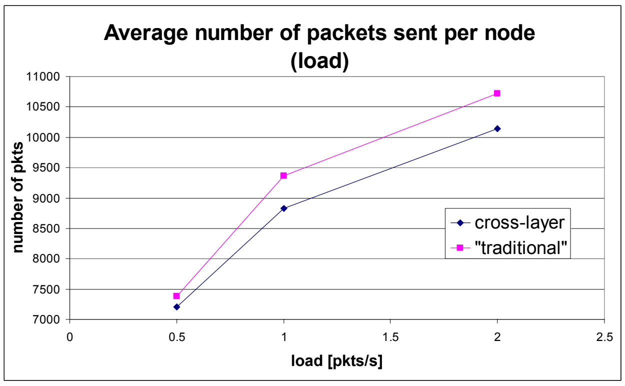

Fig. 9. Average number of packets sent per node

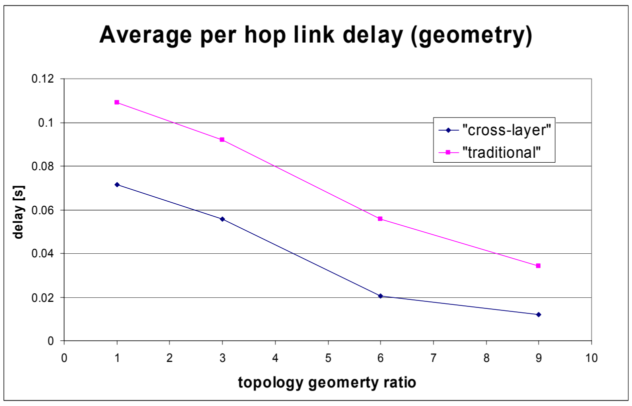

Fig. 10. Average per hop delay

Fig. 9 shows the load reduction effect. As can be seen, as the load increases the load reduction effect grows. In this particular case we only have an average saving of up to 5%. The maximum saved amount was 15% for the topology geometry scenarios discussed later on. This shows that, although not the primary goal, our reference application is able to reduce the load.

Fig. 10 presents the improvements of the per hop delay, which is the time from a packet being received at one hop till it is received at the next hop. In almost all tested scenarios the delay performance of our cross-layer approach was better than with the "traditional", layered approach, the only exception being highly dense networks (however, there was still a significant load balancing effect). The maximum delay improvement measured was 65%.

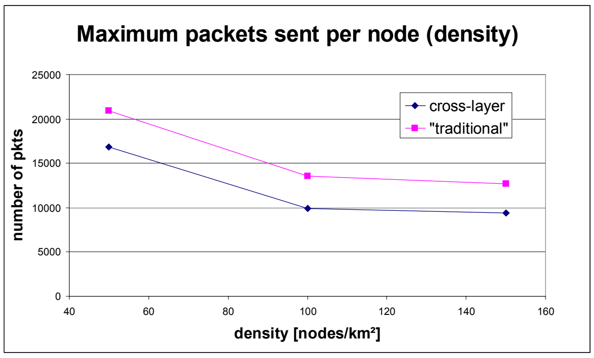

Fig. 11. Maximum packets sent per node

Bottlenecks in the network are most likely the first ones to fail due to the depletion of their batteries. They also are a cause for regions with high collision rates and delay. Fig. 11 depicts the reduction of the load at the worst bottleneck nodes in the network. It shows the independence of the network density on this effect. In this particular scenario the reduction at the most overloaded node goes up to 25%. This load relief effect at highly overloaded nodes was observed in all tested scenarios.

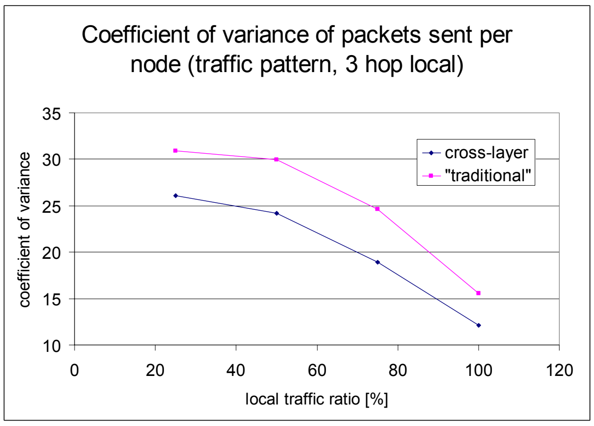

Fig. 12. Coefficient of variance

The last performance metric for the evaluation of our load balancing reference application is the coefficient of variance of the average packets sent per node during a simulation run. We can't use the simpler standard deviation here, since with our algorithm we send fewer packets per node, which makes a direct comparison of the standard deviations impossible. In Fig. 12 the coefficient of variance is show dependent of the local traffic ratio, which in this case is the amount of traffic to destinations within 3 hops. In the tested scenarios the coefficient is up to 20% smaller using our cross-layer approach and can be observed in all scenarios.

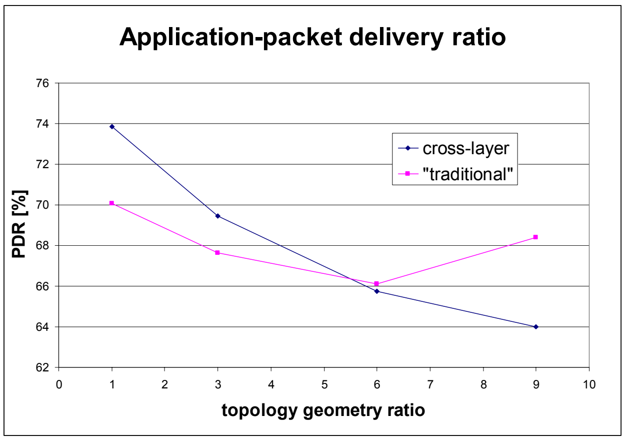

Fig. 13. Packet delivery ratio

In our tested mobility scenario our cross-layer architecture was only slightly outperforming the "traditional" approach. This is due to the fact that with random mobility load balancing comes as a side effect since nodes traverse differently loaded areas constantly. Still, our algorithm achieves a coefficient of variance of 7.92 compared to 8.61 with the layered architecture and an average maximum load of 8941.5 compared to 9564.55.

In only one scenario the PDR dropped slightly below the reference PDR of the layered protocol stack. The PDR of the cross-layer approach was 64 compared to 68.4 in the topology with an aspect-ratio of 1:9 (see Fig. 13). This phenomenon can be explained by the fact that in topologies with such an aspect-ratio, or an even higher one, there are only few paths through the network. Those are extremely overloaded compared to nodes at the edges of the network, for example. Nodes on these paths most likely operate beyond the drop bound.

V. Conclusion & Future Work

The deployment of TCP in wireless ad-hoc environments made it very clear that simply adopting technologies from infrastructure-based networks into highly dynamic network environments can be very inefficient. Cross-layer designs are a promising new paradigm to provide an architectural framework for more efficient network stacks, especially for wireless networks.

We presented CrossTalk, a generic architecture for cross-layer optimizations. The novel feature of our system is the ability to reliably establish a network-wide, global view of the network of one or multiple metrics. Having such a global view of the network a node can use that information for local decision processes. We thoroughly analyzed the global view approach to make sure it is suitable for the complex system dynamics of ad-hoc networks.

We also exemplary applied our architecture to a load balancing algorithm. With this algorithm we were able to reduce the per hop packet delay up to 65%. We also could relief the bottleneck nodes from up to 25% of their packet load. Additionally, we could reduce the coefficient of variance of the packets send per node by up to 20%.

In the future we plan several improvements to our architecture. For example we want to dynamically find out when we should and when we shouldn't piggyback information to further reduce the overhead of the data dissemination process. For the same purpose we want to restrain information exchange to clusters.

We also want to improve the presented load balancing algorithm. We plan to dynamically set the thresholds e.g. from evaluating the standard deviation of collected samples or react to changing path loads. In addition we will test more aggressive strategies to further improve the algorithm. Finally, we want to evaluate the system again with more realistic mobility models such as an attraction point mobility model.

References

[1] D.B. Johnson, J.-P. Hubaux, "Report on the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)", in *Mobile Computing and Communicaitons Review*, Volume 6, Number 3, 2002

[2] I. Stojmenovic, "Position-Based Routing in Ad Hoc Networks", In *IEEE Communications Magazine*, July 2002

[3] S. Shakkottai, T.S. Rappaport, P.C. Karlsson, "Cross-layer Design for Wireless Networks", in *IEEE Communications Magazine*, October 2003

[4] A. Safwat, H. Hassanein, H. Mouftah, "Optimal Cross-Layer Designs for Energy-Efficient Wireless Ad hoc and Sensor Networks", *IEEE IPCCC 2003*

[5] V. Kawadia, P.R. Kumar, "Principles and Protocols for Power Control in Wireless Ad Hoc Networks", in *IEEE Journal on Selected Areas in Communications*, January 2005

[6] Q. Zhang, W. Zhu, Y.-Q. Zhang, "A cross-layer QoS-Supporting Framework for Multimedia Delivery over Wireless Internet", in *International Packetvideo Workshop 2002*

[7] J. Chen, T. Lv, H. Zheng, "Joint Cross-layer Design for Wireless QoS Content Delivery", *IEEE International Conference on Communication 2004*

[8] U.C Kozat, I. Koutsopoulos, L. Tassiulas, "A Framework for Cross-layer Design of Energy-efficient Communnication with QoS Provisioning in Multi-hop Wireless Networks", *IEEE INFOCOM 2004*

[9] M. Conti, G. Maselli, G. Turi, S. Giordano, "Cross-Layering in Mobile Ad Hoc Network Design", in *IEEE Computer Magazine*, February 2004

[10] M. Issoufou Tiado, R. Dhaou, A.-L. Beylot, "UCL: A new Method for Cross-Layer Network Modelling", *Technical Report IRIT (2005-1-R) 2005*

[11] H. Ritter, R. Winter, S. Schiller, "A Partition Detection System for Mobile Ad-Hoc Networks", *IEEE SECON 2004*

[12] V. Kawadia, P.R. Kumar, "A Cautionary Perspective on Cross Layer Design", in *IEEE Wireless Communication Magazine*, February 2005

[13] Y.C. Tseng, S.-Y- Ni, Y.-S. Chen, J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Networks", *IEEE/ACM MOBICOM 1999*

[14] B. Zhou, A. Marshall, J. Wu, T.-H. Lee, J. Liu, "A Cross-layer Route Discovery Framework for Mobile Ad Hoc Networks", *Technical Repor*t, 2004, available at http://www.ee.qub.ac.uk/dsp/research/telecomms/research/PRSM/Publication/archive.htm

[15] C. Maihöfer, T. Leinmüller, R. Eberhardt, "Improving the Usable Capacity of Ad Hoc Networks", *KiVS 2005*

[16] S.-J. Lee, M. Gerla, "Dynamic Load-Aware Routing in Ad hoc Networks", *IEEE ICC 2001*

[17] H. Hassanein, A. Zhou, "Routing with Load Balancing in Wireless Ad hoc Networks", *ACM MSWiM 2001*

[18] B.-J. Kwak, N.-O. Song, L. E. Miller, "On the Scalability of Ad Hoc Networks: a traffic analysis at the center of a network", *IEEE WCNC 2004*

[19] H. Lundgren, E. Nordström, C. Tschudin, "Coping with Communication Gray Zones in IEEE 802.11b based Ad hoc Networks", *WoWMoM 2002*