2. Background

The first part of this chapter briefly introduces several design alternatives to the cross-layer design. It evaluates their advantages and disadvantages when applied to mobile ad hoc networks. The second part thoroughly describes the fundamental properties of the cross-layer paradigm and explains its particular applicability in dynamic environments.

2.1. Design Alternatives

The probably most basic design option for a network architecture is to have one monolithic stack comprising the full network functionality such as reliable transport, addressing and routing to only name a few. Architecturally being extremely simple, it is clear that there are many disadvantages making a monolithic solution unattractive. Software complexity issues are one of the main challenges and therefore monolithic systems might be deployed if at all to extremely resource constrained systems (e.g. sensor nodes) or in systems where most functions are carried out by hardware rather than software. Other disadvantages become clear when compared with the second architectural choice, the layered protocol stack.
Layered protocol stacks as used in almost all modern communication systems have a number of key characteristics that make them the primary architectural choice when it comes to designing a new network system. Sharing the same architectural framework, those systems differ in number, behavior and possibly function of the different layers. The underlying concept of layer separation is intuitive and simple. Each layer has a well defined functionality making them independent of each other. Such a function could be the routing of a message over multiple hops from a source node to some destination node(s) through the network for example. The routing layer would be responsible for performing anything that has to do with the task of determining the most appropriate path or next hop for a packet. That could include such operations as routing table maintenance, route optimization and addressing. This functional encapsulation allows separate operations to be offered to upper layers as a service. With this approach the level of abstraction grows towards upper layers as they build upon

lower layers and can offer a higher degree of flexibility on the service they provide. The highest degree of flexibility is offered by the topmost layer, the application layer, as an application can have any arbitrary function.
Layer interaction is reduced to service invocation through well defined interfaces. Data that is passed from one layer to the next is not interpreted but instead treated as payload. Each layer adds header and/or trailer information containing control information. As a packet travels through the network the control information found in the headers are interpreted at the corresponding layers on its way to the final destination. In other words the network layer header is only interpreted at the network layer of other nodes. This way only peer layers communicate directly with each other to perform their function.
There are several advantages a layered approach has as opposed to a purely monolithic alternative. The overall design complexity is greatly reduced. Since a layer has to perform a certain function, the design effort can only be focused on that single function without concerning upper or lower layers. Teams of developers only need to define interfaces so that the service(s) a layer is providing can be accessed by upper layers. The modularity, the layers provide, allows for potential arbitrary combination of protocols. From an architectural point of view it also improves the maintainability as a new version of a protocol can be inserted without having to alter the rest of the network stack. This of course can be more complicated in a real implementation where the functionality might be deeply embedded into the operating system, for example. Still, a layered network stack is advantageous in many respects. Furthermore, the longevity and widespread use of this kind of reference model demonstrates its benefits and generic applicability.
A third design alternative, which is only rarely found, can be seen as a combination of the monolithic and layered approach. Two or more functions are combined in a single layer which potentially would be separated by several layers in a traditional protocol stack. A good example for this approach is MADPastry [4]. MADPastry combines network layer routing with an application layer peer-to-peer substrate based on a distributed hash table (DHT) to efficiently realize a key-based routing overlay for ad hoc networks. This combination still provides the functions of both concepts only in a much more efficient and modified way but at the same time layer separation is given up to some degree.

2.2. DynaMO: An Example for In-layer Adaptations

The most intuitive approach to counteract the various ad hoc network system dynamics would be to tailor protocols and applications towards the special circumstances found in those environments on a per-protocol or per-application basis. The fact that there are already many protocols explicitly designed for ad hoc networks is a good indicator for the intuitiveness of the approach. At the same time it shows that there is no easy, generally applicable solution. Protocols are simply not adaptive enough to operate in a broad spectrum of scenarios and therefore many alternative protocols were designed, many of which are tailored towards special networking environments. Most of them do not have a directly comparable functional counterpart in infrastructure based networks. Therefore, an application called Dynamic Mobility-Aware Overlay, short DynaMO ([5][6][7][8][9]), was designed based on an existing system for the Internet but

explicitly modified to be suitable for the special needs found in ad hoc networks. This way DynaMO serves as a good example for the process of protocol and application adaptation for ad hoc networks as it can directly be compared against the Internet version of the application. DynaMO is also a good and interesting case study as it serves as a peer-to-peer (P2P) substrate which can potentially support other services found in today's Internet such as DNS and file sharing. As these P2P systems share a good number of key characteristics with ad hoc networks they seem to be suitable to be deployed in environments that are distributed and dynamic in nature. These characteristics include the self-organizational properties, the decentralized operation mode and resilience and robustness in the face of the failure of participants.

DynaMO is a peer-to-peer system or overlay network based on a distributed hash table (DHT). To be more precise, DynaMO is based on Pastry [10][11] which has been used to implement several Internet applications on top of it such as group communication [12][13] or storage systems [14][15]. A DHT comprises an extremely large ID space (typically in the order 2^{120} to 2^{160}). Every node participating in the overlay network needs to assign itself an ID within that range. This is done by hashing some unique node specific value such as the medium access control (MAC) address using a common hash function used by all participating nodes. Due to the extremely large ID space collisions are avoided, i.e. the assignment of two identical overlay IDs to different nodes in the network. Objects that are added to the overlay network are also assigned an ID from the same ID space using the same hash function. In Pastry, the node with an ID numerically closest to an object's ID is responsible for that given object. This scheme might be different for other DHTs but a unique assignment from node ID to object ID range is fundamental to all DHTs. The basic operation that each DHT provides is `lookup(key)→node`, i.e. the DHT is able to locate the node that is responsible for a given key within the network or in other words a DHT provides a mechanism for key-based routing. The most significant properties of Pastry and other DHTs are the upper bound given for the amount of routing state needed to guarantee routing convergence and the total amount of hops in the overlay network until the node responsible for a given key is reached. Both upper bounds are typically in an order of O(log n) where n is the number of participating nodes.

The way Pastry achieves this efficiency is quite simple. In every routing step Pastry increases the matching prefix length between the lookup key and the next node in the routing process by at least one until the final destination node is reached. To be able to do this, a Pastry node maintains a small routing table that maps overlay IDs to network addresses. The table consists of b columns and d rows where b is the base of the overlay ID and d is the ID's number of digits. As an example a Pastry DHT routing table with 120 bit hexadecimal IDs would have 16 columns and 30 rows. Each row r_i with $i \in [0,d)$ contains nodes with IDs that have a matching prefix length of i. For the first row (r_0) that would mean that the IDs in that row have no matching prefix at all and for the last row (r_{d-1}) all but the last digit match the key. The columns c_i with $i \in [0,b)$ are organized in a way that the digit following the matching prefix is equal to i. In a routing process the key lookup message is forwarded to a node in the routing table that has the longest matching prefix. Additionally, every node maintains a set of nodes that are the numerically closest nodes called the leaf set. The leaf set contains l nodes

with *l/2* nodes that are the numerically closest nodes bigger than the node's ID and *l/2* nodes which are the numerically closest but smaller than the node's ID. This information is needed at a node, so that it is able to conclude whether it is the final destination or not.

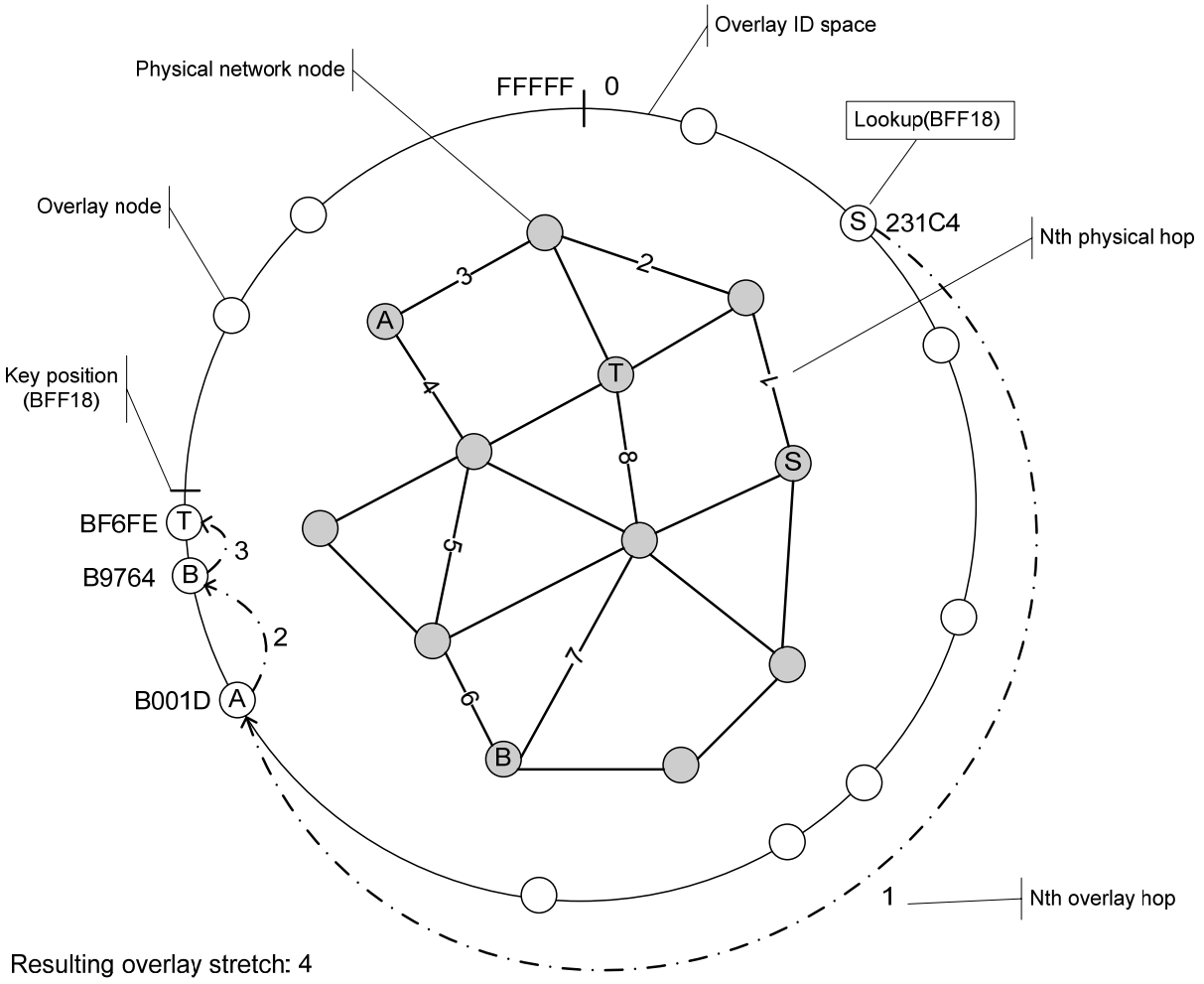

Fig. 2.1 Overlay lookup example

In addition to the similar characteristics, the small footprint and the efficiency of Pastry suggests that it is a very good system to be deployed in ad hoc networks as they are extremely resource constrained. But it turns out that there is one intrinsic problem with Pastry and other DHTs that make them unsuitable for wireless multi-hop environments. The routing bound of O(log n) described above is only applicable to the overlay routing process, i.e. the bound only concerns the virtual links in the overlay network. Each of these virtual links typically consists of multiple hops in the real physical network. Consider a key-based routing process on the Internet as displayed in Fig. 2.1. A source node S tries to route towards a key k. The routing process might go from node S located in Germany to the USA from there to Japan back to somewhere Europe and so forth just to end up somewhere in Germany again where the target node T is responsible for k. On the overlay network this process seems to be efficient as only a few virtual links are traversed to find node T but in the real physical network the message criss-crossed the Internet several times inducing a lot of traffic. This discrepancy between the physical accumulated path length of the overlay lookup process and

the direct physical path length to the destination, in the example above node *T*, is called the *overlay stretch*.
The overlay stretch might not be a problem in infrastructure-based networks such as the Internet since routing is a very efficient and fast process but in mobile ad hoc networks routing is much more problematic. Nodes are mobile making routes instable which in turn will reduce the packet delivery ratio. The longer a route is the higher is the probability of message loss due to topological changes. Such a route failure can trigger broadcast based mechanisms to repair the route or find a completely new one congesting the network. Therefore, long routes should be avoided. Additionally, due to the unreliable medium messages will fail due to collisions on the medium and due to bit errors in the packets cause by interference. In summary this means that the overlay routing efficiency in ad hoc networks is not the primary concern but the total length of the physical path traveled during a lookup process must be minimized. In other words, the overlay stretch must be as small as possible for a lookup process to successfully deploy DHTs in ad hoc networks.
DynaMO was designed to reduce the length of the overlay stretch. It exploits physical locality in the construction of the overlay network which translates into an optimized overlay stretch. In a DynaMO network, nodes that are physically close are also likely to be numerically close on the overlay which in Pastry networks only rarely happens and only due to chance. DynaMO has two mechanisms to achieve physical locality whereas only the random landmarking approach will be discussed here. DynaMO's approaches differ primarily from Pastry's approach by the way in which overlay IDs are assigned. Pastry's overlay construction basically works in a top-down fashion, i.e. Pastry randomly assigns overlay IDs regardless of the underlying physical topology. This is reflected in the total randomness of the spatial overlay ID distribution as can be seen on the left side of Fig. 2.2. Subsequently it tries to make the physical proximity fit into the overlay routing state through the join process and overlay routing table maintenance. Proximity is achieved by choosing the physically closest node among candidate nodes for a routing table entry. Since with a growing matching prefix length the amount of candidate nodes is significantly reduced, the probability of an overlay hop inducing a large amount of physical hops increases significantly with the increasing matching prefix. That means that the last hop of a lookup process is most likely the longest whereas the first one is most likely the shortest.
In contrast, DynaMO constructs the overlay network in a bottom-up fashion, i.e. the overlay is built considering locality information from the underlying network. Before a node joins the overlay, it gathers information concerning its physical neighborhood and uses it to assign itself an appropriate overlay ID. One way to do this is random landmarking. Conventional landmarking, as introduced in [16] and [26], suffers from the limitation that it assumes a set of fixed, stationary landmark nodes such as the DNS root servers. All overlay nodes are expected to know the landmark nodes and to measure their respective distances to those landmarks. This, obviously, reintroduces the client-server concept into the bootstrap process. Especially in networks where nodes are expected to fail or leave frequently, as it is the case for ad hoc networks, there are usually no sets of fixed nodes available, which renders this approach infeasible. Therefore, DynaMO introduces random landmarking (RLM) into the overlay construction

process. RLM utilizes the overlay lookup capabilities to locate overlay nodes responsible for a fixed set of landmark keys (overlay IDs). These nodes serve as temporary landmarks for a joining node. It is important to understand that the keys have to be chosen in a way that they divide the overlay ID space into equal portions. For example, in a network with an ID base of 16, an appropriate set of landmark keys would be: 000..00, 100..00, 200..00, . . . , F00..00. The joining node then measures the distances to those temporary landmarks and assigns itself an ID based on its landmark ordering. The advantage of this approach is that "landmark nodes" can fail and others will simply step in as Pastry will automatically redirect future key lookups to those nodes now responsible for the landmark keys. After having measured its landmark distances, the joining node adopts an ID prefix of a certain length from the landmark node closest to itself. The ID remainder can be assigned randomly or can be based on an algorithm that further takes into account the physical neighborhood. The length of the ID prefix that the new node shares with its closest landmark node can be determined using the following formula:

$$prefix\ length = \lfloor \log_b k \rfloor$$

where b is the ID base and k the number of landmark keys. As can be seen, the number of landmark keys should preferably equal to a power of b.

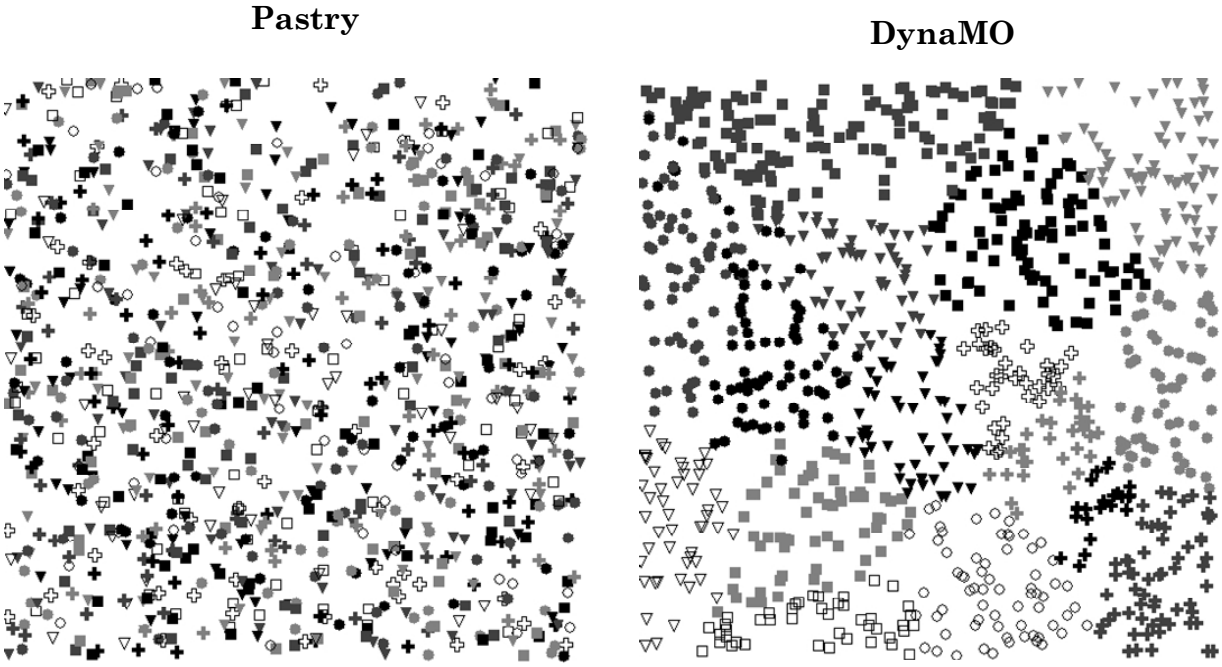

Fig. 2.2 Spatial prefix distributions as generated by Pastry and DynaMO. Equal symbols and colors represent equal prefixes.

This approach has the following effects. First of all, it leads to physically close nodes forming regions with common ID prefixes as shown on the right side in Fig. 2.2., which means these nodes are also likely to be numerically close to each other in the overlay ID space. This, in turn, leads to the desired effect that nodes having a certain matching prefix length in the routing state are likely to reference physically close nodes. This way the closer a routing process is approaching the lookup key and therefore a prefix cluster, the shorter the overlay

hops will be in terms of induced physical hops. Note that there are still less and less candidate nodes to choose from to fill a certain overlay routing table entry as the row number increases, but with DynaMO the likelihood of these candidates being physically close to the current node also increases from row to row.
RLM needs to deal with mobility-induced topology changes as they lead to disintegration of the ID clusters. Every node periodically re-measures its distance to the current landmark nodes. If its ID prefix is still congruent with the prefix of its closest landmark node, it uses some coarse grained mechanism to adjust the re-measurement interval by some factor. Otherwise, it will re-assign itself a new overlay ID based on the same strategy as used during its bootstrap and will rejoin the network under its new ID.
There are more mechanisms to make the landmarking process more efficient but the general description of the process should suffice here to give an idea how applications and protocols must be changed in order to make them work in ad hoc networks. These, what we call in-layer adaptations are only one aspect of the protocol design, which by its own is not sufficient but eventually necessary.

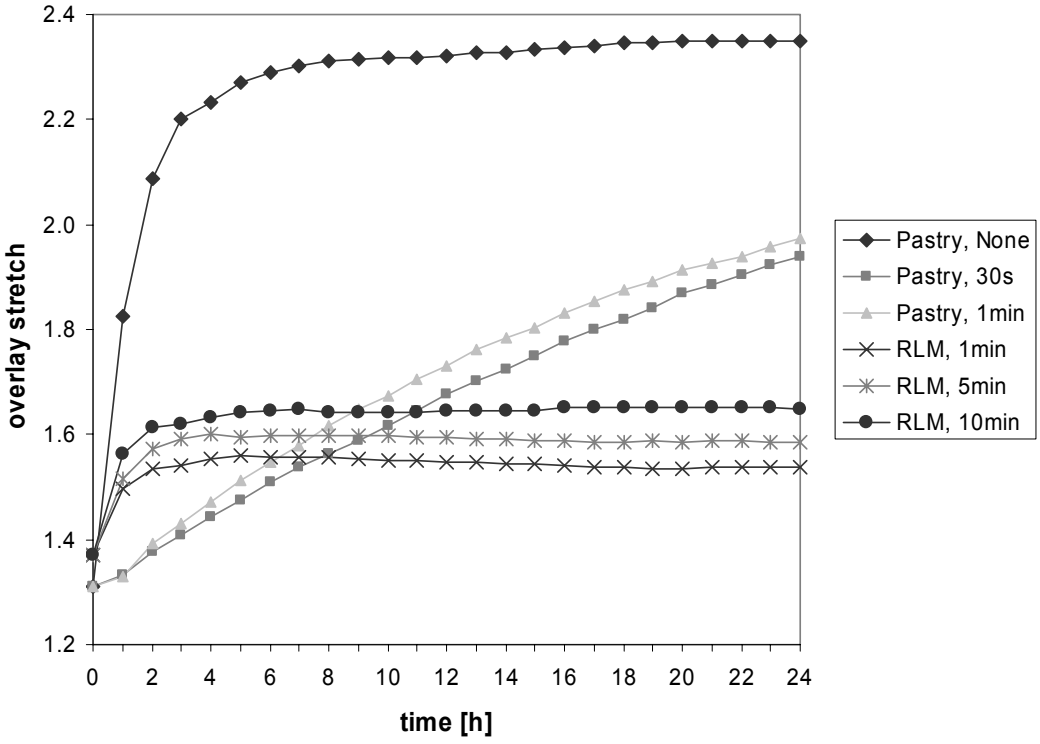

Fig. 2.3 Overlay stretch change over time with Pastry and RLM

DynaMO was tested in a vast amount of ad hoc networking scenarios and was always compared against Pastry's performance (for a complete set of results refer to appendix B). Using RLM in static scenarios it was possible to drop the overlay stretch significantly compared to Pastry. Using the same optimization techniques Pastry utilizes it was even possible to drop the overlay stretch below Pastry's theoretical optimal performance. Looking at the same overlay stretch performance, DynaMO was far more lightweight than Pastry by up to a factor of 7. This clearly shows the superiority of DynaMO. It outperforms Pastry in the two key factors mentioned before. Firstly, the shortened overlay lookup lengths reduce the probability for packet loss due to route changes, collisions and

interference. Secondly, the overhead is reduced when comparing equal overlay stretches of both Pastry and DynaMO, which causes the collisions and interference in the first place. In mobile network scenarios DynaMO also clearly outperforms Pastry. Pastry is not able to maintain a good overlay stretch, even when it utilizes its routing state optimization technique as can be seen in Fig. 2.3. The overlay stretch degrades significantly over time. Even with a routing table maintenance interval of 30 s and 1 min in low mobility networks (average nodal speed of 0.6 m/s) the overlay stretch quickly degrades. DynaMO on the other hand is able to stabilize the overlay stretch in the face of node mobility at an excellent low level depending on the landmark key re-measurement interval used to maintain the prefix clusters.

DynaMO is a good example of in-layer adaptations to tailor applications or protocols towards the special circumstances of ad hoc networking environments. What DynaMO also clearly shows is that in-layer adaptations are necessary but they are only partially able to overcome the problems found in ad hoc networks as [9] shows. The design of new protocols and applications and the re-design of those found on the Internet is therefore only one aspect of their successful integration into dynamic environments. The two main problems that remain are the ability to adapt to existing and to changing networking conditions on the one hand and a mechanism to let a node evaluate its own status against the network-wide status according to some metric such as mobility or energy. Let us consider DynaMO as an example. Obviously the stability of the ID prefix clusters depends on the re-measurement interval of the DynaMO nodes which counteracts the disintegration of the clusters due to the mobility of the participants. DynaMO uses a fixed re-measurement interval which is set prior to deployment. To find the optimal re-measurement interval is difficult. Prior to joining a network, a metric for the average topological link change rate must be known to do this which might not be a realistic assumption. But even if the interval can be set to an optimal value at deployment time the conditions can change or the nodes might roam between different networks with different characteristics as they move. Another aspect is the choice of the node that becomes a temporary landmark. A fast moving node should never become a temporary landmark as it could result in many ID changes which would generate a huge amount of control messages. In a scenario where cars, pedestrians and others participate in the overlay network, a fast moving vehicle should assign itself an ID relatively far away from a landmark ID. But how does a node know that it is relatively fast? If it is a car it might use its speedometer. But if the whole network consists of cars it still does not know whether it is moving fast in comparison with the others. What is needed is some global knowledge about the topological mobility of nodes so that each node can estimate its relative mobility compared to the rest of the network. A different example would be energy. A landmark node has to handle more traffic as compared to none-landmark peers since it has to handle measurement traffic. Nodes that have relatively good energy resources should then assign themselves an ID closer to a landmark key as they will then most likely become a landmark node or take over the role of a landmark in case the current landmark fails. Again, some form of global knowledge is needed to be able to locally decide whether the local energy resources are good as compared to the rest of the network.

2.3. Cross-layer Design

As already mentioned layered architectures have certain desirable properties. Other than the reduced design complexity, such architectures exhibit a high degree of modularity which allows an easy replacement and theoretically an arbitrary combination of protocols. This is important for rapidly updating or replacing protocols. It also plays an important role in maintaining a system. Errors can be tracked back fast and efficiently as they can easily be associated with a certain protocol. These properties are predominantly of importance in commercially operated, large-scale, infrastructure-based, centrally administrated networks. In those kinds of networks, performance is usually increased by hardware means whereas potential increases through architectural alterations are not considered attractive enough or too cumbersome to a certain extend. A changed architecture could also potentially be less stable and secure and would also need some time to reach a sufficient maturity. In addition, these networks do not suffer from such rapid changes in the networking environment, coordination problems and intrinsic bottlenecks as found in ad hoc networks. That means that in reliable, high performance, infrastructure-based networks, there is no fundamental need to switch to a new network architecture that is able to alleviate the problems mentioned above.
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, a more flexible approach is promising since the possible performance gains can significantly improve the scalability, the delay performance and the throughput. The solutions to these issues are the cornerstones to make ad hoc networks attractive for an actual deployment and a ubiquitous use in the future.
Cross-layer approaches are the most promising design models to serve as a blueprint for dynamic network architectures. The concept behind cross-layering is rather intuitive. Instead of treating a layer as a completely independent functional entity, information can be shared amongst layers. This information can be used to adapt protocol functionality in the presence of changing networking conditions, for decision processes such as route selection and as input for algorithms. Through sharing information, mechanisms of different protocols can be combined such as network layer topology maintenance and physical layer power control for example. It is even possible to create new kinds of adaptive applications such as multimedia applications which are sensitive to changing networking conditions. The ability to share information across layers is the central aspect of cross-layer design. So instead of a mere replacement, cross-layering can be seen as an enhancement of the layered approach. The ultimate goal is to preserve the aforementioned key characteristics of a layered architecture and in addition to allow for performance improvements and a new form of adaptability.

Fig. 2.4 illustrates the general concept of cross-layer design. It shows possible information that can be shared such as topology information at the network layer generated from the routing state or information about received signal strength form the lowest layer. The figure also shows a possible adaptation at the network layer. The path selection process could be enhanced by the information provided by the other layers. QoS requirements of an application could determine the most suitable path amongst a number of candidates or the bit error rate could be used to determine the most reliable next hop. Considering DynaMO as an example the topology information provided by a network layer protocol could be used for the ID assignment process, as a landmark ID should preferably be assigned to a node that is well connected (a large amount of one-hop neighbors) to prevent disconnections.

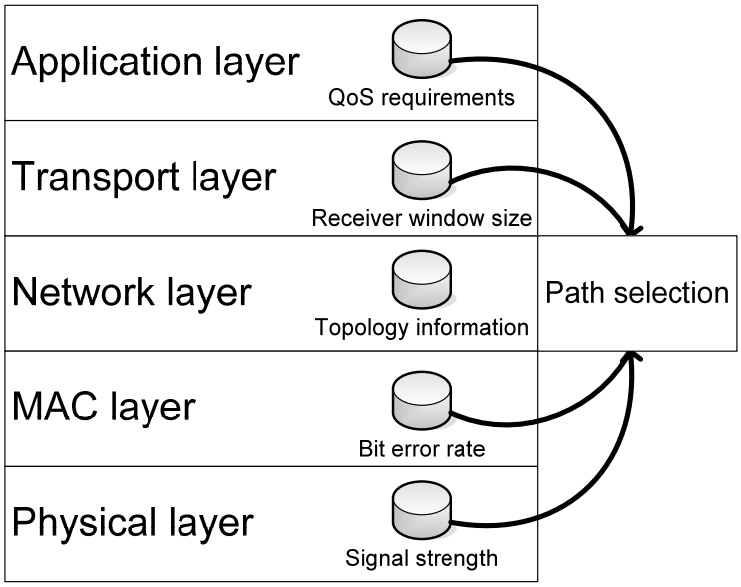

Fig. 2.4 Possible cross-layer adaptations at the network layer

As only being a framework, a cross-layer architecture should not actively steer protocol behavior or influence the state of a protocol or application actively. It should instead only provide certain services to protocols and applications. These services comprise everything that has to do with cross-layer data management such as gathering of the data, event notification on data change, data composition and formatting and providing interfaces for data access. The active adaptations should only occur within the protocols and applications themselves as this helps to preserve the properties of the layered approach.

As ad hoc networks only start to emerge, introducing cross-layer concepts into the network architecture should be done simultaneously to let them evolve together. Right from the start protocol designers should include cross-layer capabilities so that the general cross-layer concept can be tested and mature over time.

As already mentioned, cross-layer approaches are more adaptive compared to purely layered architectures. From an architectural view, they are also preferable to approaches such as MADPastry that combine functionality from different layers into one functional entity. This is due to the fact that this approach is highly complex when many functions are combined. A question that arises is how to identify functions that should be combined and those that shouldn't. This kind

of approach might also lead to frequent updates as new protocols are introduced. Ultimately, it might lead towards a small layered architecture where a layer exposes the drawbacks of a monolithic stack.

2.4. Summary

Amongst the existing design alternatives for network protocol stacks the layered approach is the de facto standard adopted by almost all existing networks. Its simplicity and modularity are the key reasons for its widespread adoption. To make layered protocols work in dynamic networked environments in-layer adaptations are the first step to a successful deployment of existing protocols in ad hoc networks. Good cross-layer design does not replace the layered approach but weakens the strict layering to some degree by allowing protocols to share information. In other words, cross-layer design can be seen as an enhancement of a layered protocol stack with the goal to preserve its characteristics and provide a mechanism for performance improvements and adaptations at the same time.