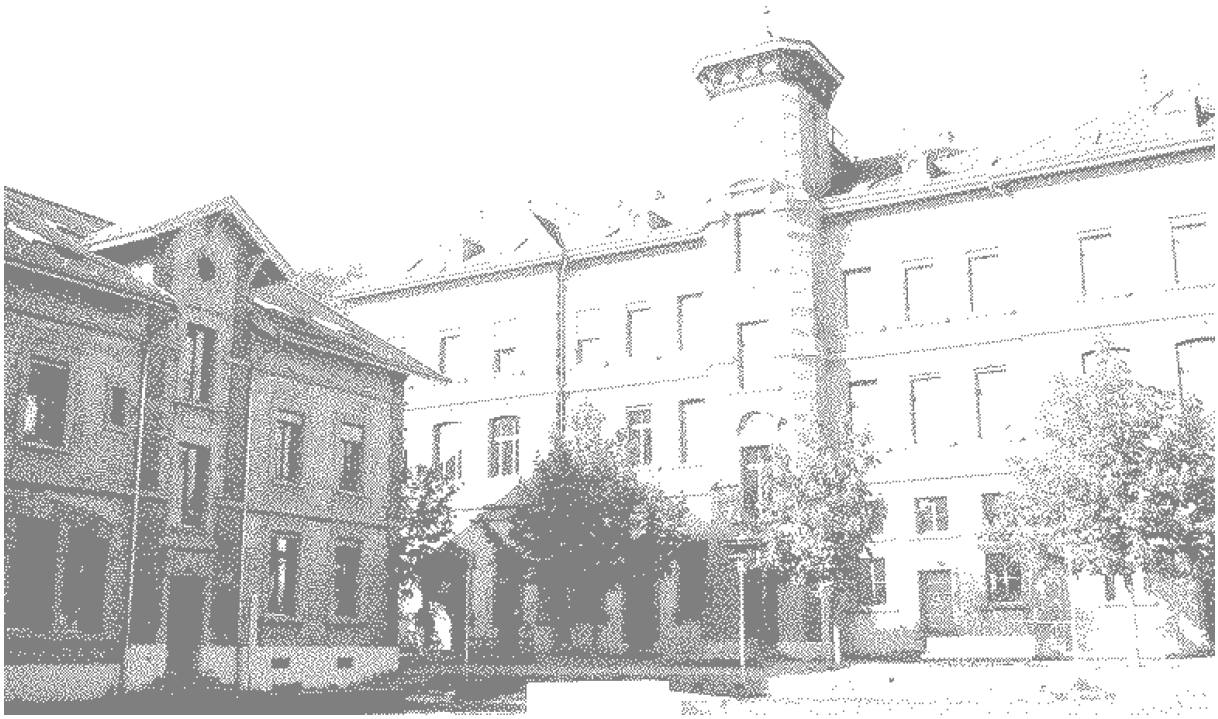


BeNeFri Summer School 2009 on Dependable Systems

Marc Brogle, Torsten Braun (eds.)

Technical Report IAM-09-006, September 8, 2009

Institut für Informatik und angewandte Mathematik, www.iam.unibe.ch

BeNeFri Summer School 2009 on Dependable Systems

Marc Brogle, Sabina Serbu, Dragan Milic, Markus Anwander, Philipp Hurni, Christian Spielvogel, Claire Fautsch, Derin Harmanci, Lucas Charles, Heiko Sturzrehm, Gerald Wagenknecht, Torsten Braun, Thomas Staub, Carolin Latze, Ronny Standtke

Technical Report IAM-09-006, September 8, 2009

CR Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design; C.2.2 [Computer-Communication Networks]: Network Protocols; C.2.3 [Computer-Communication Networks]: Network Operations; C.2.4 [Computer-Communication Networks]: Distributed Systems

General Terms:
Design, Management, Measurement, Performance, Reliability, Security, Evoting

Additional Key Words:
peer-to-peer, wireless mesh networks, wireless sensor networks, overlay multicast, network security, anonymity, transactions, evoting

Institut für Informatik und angewandte Mathematik, Universität Bern

Abstract

The BeNeFri Summer School 2009 on Dependable Systems in Schloss Münchenwiler from June 10-12 was organized by four research groups from the Universities of Bern, Fribourg and Neuchâtel. The University of Bern was represented by the research group “Computer Networks and Distributed System” of the Institute of Computer Science and Applied Mathematics at the University of Bern, headed by Prof. Torsten Braun. The research group “Telecommunications, Networks, Security” of the Department of Computer Science, headed by Prof. Ulrich Ultes-Nitsche, represented the University of Fribourg. The University of Neuchâtel was represented by two research groups from the “computer science department (IIUN)”, namely “distributed systems” headed by Prof. Peter Kropf and “dependable systems and networks” headed by Prof. Pascal Felber. The focus of this retreat was to present and discuss recent research results and currently ongoing research activities of the members of the four research groups. The research group members gave twenty-two presentations, from the areas of overlay networks, wireless mesh and sensor networks, network security, distributed systems, zero-proof knowledge, transactions and security in e-voting. Extensive time (typically 45 minutes per talk) has been allocated to allow detailed presentations and discussions. This technical report summarizes the various talks and describes mostly unpublished work that is currently in progress.

Contents

1 Quality of Service for ”NICE” Overlay Multicasting **1**
- 1.1 Introduction . . . 1
- 1.2 NICE Overlay Multicast . . . 2
- 1.3 QoS for NICE Overlay Multicast . . . 3
- 1.4 Evaluation . . . 4
- 1.5 Normal NICE and QoS Enabled NICE . . . 5
- 1.6 NICE using Soft QoS . . . 5
- 1.7 NICE with Node to Root RTT Constraints . . . 6
- 1.8 Conclusion . . . 7

2 Hypercube-like P2P Overlay for Flexible Routing **9**
- 2.1 Introduction . . . 9
- 2.2 Discussion . . . 10
  - 2.2.1 Overlay Structure . . . 10
  - 2.2.2 Overlay Routing . . . 11
- 2.3 Conclusion and Outlook . . . 13

3 NetICE9 **15**
- 3.1 Introduction . . . 15
- 3.2 VIVALDI’s Instability . . . 16
- 3.3 Reasons for VIVALDI’s Instability . . . 16
- 3.4 NetICE9 . . . 18
  - 3.4.1 Crystallization . . . 18
  - 3.4.2 Choice of Neighbors . . . 19
- 3.5 Evaluation . . . 21
- 3.6 Conclusion . . . 21

4 Energy efficient Multihop Linklayer Protocol **25**
- 4.1 Introduction . . . 25
- 4.2 Discussion . . . 25
- 4.3 Conclusion and Outlook . . . 27

5 Adaptivity of Energy-Efficient MAC Protocols **31**
- 5.1 Introduction . . . 31
- 5.2 Discussion . . . 31
  - 5.2.1 Simulation Settings . . . 32
  - 5.2.2 Energy-Throughput and Energy-Latency Tradeoffs . . . 33
- 5.3 Conclusion and Outlook . . . 35

6 Scalable Video Streaming to Mobile Devices 37
6.1 Introduction . . . 37
6.2 Temporal Scalability . . . 38
6.3 Spatial Scalability . . . 39
6.4 Scalable Coding: Signal-to-Noise-Ratio Scalability . . . 41

7 Challenges in Domain Specific Information Retrieval 43
7.1 Introduction . . . 43
7.2 Discussion . . . 43
7.2.1 Social Science . . . 44
7.2.2 Blogsphere . . . 45
7.2.3 Biomedicine . . . 46
7.3 Conclusion and Outlook . . . 47

8 Testing and evaluating Distributed Algorithms 49
8.1 Introduction . . . 49
8.2 Discussion . . . 49
8.2.1 Challenges from pseudocode to implementation . . . 49
8.2.2 teDA framework . . . 51
8.2.3 Challenges addressed by teDA . . . 52
8.3 Conclusion and Outlook . . . 54

9 Transactional memory applied to Application Server 55
9.1 Introduction . . . 55
9.2 Discussion . . . 56
9.3 Results . . . 57
9.4 Conclusion and Outlook . . . 58

10 Event Stream Processing meets Software Transactional Memory 61
10.1 Introduction . . . 61
10.2 Discussion . . . 61
10.2.1 Requirements . . . 62
10.2.2 Enhanced Component . . . 62
10.2.3 Evalution . . . 63
10.3 Conclusion and Outlook . . . 64

11 Reliable Multicast in IP-based Wireless Sensor Networks 67
11.1 Introduction . . . 67
11.2 Designing Multicast in WSNs . . . 67
11.2.1 Overlay Multicast . . . 68

11.2.2 Reliability 69
11.2.3 Implementation and Evaluation 69
11.3 Conclusion and Outlook 71

12 Cooperative Cognitive Context-aware Composable $(Co)^4$ Virtual Wireless Mesh Networks 73
12.1 Introduction 73
12.2 Wireless Mesh Networks 73
12.2.1 Roaming and Cooperation in Wireless Mesh Networks 74
12.2.2 Virtual Wireless Mesh Networks 75
12.3 Cognitive Wireless Mesh Networks 75
12.3.1 Cognitive Networks 76
12.3.2 Cognitive Mesh Node Architecture 77
12.4 Conclusion and Outlook 78

13 Supporting Wireless Mesh Networks during their Life Cycle 81
13.1 Introduction 81
13.2 Challenges 82
13.3 Contributions 83
13.3.1 VirtualMesh: An Emulation Framework for Wireless Mesh Networks in OMNeT++ 83
13.3.2 ADAM: Administration and Deployment of Ad-hoc Mesh networks 84
13.3.3 ViSuC: Video Support for Constructions 85
13.3.4 CTI-Mesh: Wireless Mesh Networks for Interconnection of Remote Sites to Fixed Broadband Networks (Feasibility Study) 85
13.3.5 ATOM: Adaptive Transport Over Multipaths 85
13.4 Conclusion and Outlook 86

14 EAP-TPM - A new authentication method for 802.11 based networks 89
14.1 Motivation 89
14.2 Trusted Computing 90
14.3 EAP-TPM 92
14.4 Conclusion and Outlook 93

15 Modularization of the NIO Framework and Misuse prevention in PGA 95
15.1 Modularization of the NIO Framework 95
15.1.1 Introduction to NIO 95

15.1.2 Introduction to the NIO Framework 95
15.1.3 Modularization 96
15.2 Misuse prevention in PGA 99
15.2.1 Introduction to PGA 99
15.2.2 Misuse prevention 99

1 Quality of Service for "NICE" Overlay Multicasting

Marc Brogle, University of Bern
brogle@iam.unibe.ch

1.1 Introduction

Efficient and concurrent distribution of data from one sender to multiple receivers can be achieved using the multicast paradigm. The implementation of the multicast paradigm for the Internet is called IP Multicast [1]. IP Multicast is though not widely deployed, and therefore not available for end users in the Internet today. This is mainly due to security concerns, complex provider billing agreements, configuration complexity, etc. Application Layer Multicast (ALM) [2], which normally runs on-top of Peer-to-Peer (P2P) [3] networks, offers a solution to the limited availability of IP Multicast. Improving P2P and ALM protocols by introducing Quality of Service (QoS) functionality could offer additional benefits for end users.
Our OM-QoS (Quality of Service for Overlay Multicast) framework [4, 5, 6] is a solution to enable QoS for various P2P/ALM protocols and frameworks. The OM-QoS framework enables multicast trees to support QoS regarding the construction of the paths from the multicast root node to the leaf nodes. Each node has its own QoS requirements represented as a QoS class. QoS classes can consist of a dedicated parameter, such as bandwidth, or a combination of multiple parameters that are though limited to influence only one hop (or P2P link) at a time. The following limitations apply to the QoS class construct: There is a total order relation for all QoS classes; QoS class parameters depend not on link length and number of hops in the network; there is a finite number of QoS classes. Therefore e.g. end-to-end delays over multiple hops are not considered in the QoS class construct. But a combination of bandwidth and hop-by-hop jitter is possible. The muticast tree is now built in such way that all the paths from the root to the leaf nodes have monotonically decreasing QoS requirements. Such a QoS aware tree is shown in Fig. 1.1. The thicker lines between nodes represent higher QoS requirements.
In this analysis of the QoS for the NICE [7] protocol, we do not only look at the general QoS class concept introduced by OM-QoS. We also add mechanisms to support guarantees regarding node to multicast root delays in terms of round trip time (RTT).

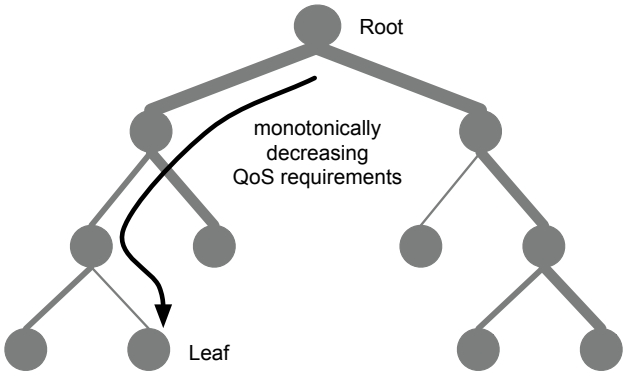


Figure 1.1: QoS Supporting Multicast Tree

1.2 NICE Overlay Multicast

NICE hierarchically arranges nodes in clusters and layers as depicted in Fig. 1.2. If nodes are in the same cluster, they are called cluster mates. Each cluster has a cluster leader, which is known by all other cluster mates. Clusters have a size between k and $3k-1$ nodes. The value k is a predefined cluster constant that is larger than zero. Cluster leaders are determined using the graph-theoretic center calculation as presented in [7]. Cluster leaders are also members of the next higher layer. Layers consists of one or multiple clusters and are ordered from the bottom layer zero to the top layer n. In the top layer, only one cluster with one node in it exists, which is then the root of the NICE network and its multicast tree.
Five invariants that have to be fulfilled at any time specify the structure of a NICE network. 1) A node is only in one cluster on each layer. 2) A node in layer L is also in layers $L-1, ..., 0$. 3) A node not in layer L can not be in any higher layer ($L+i, i \geq 1$). 4) The cluster size is between k and $3k-1$, with k being a constant larger than zero. 5) There is no more than $\log_k N$ layers and the top layer only has one node (the root).
When a node joins a NICE network, it contacts the root node, which is the cluster leader of the top layer. The root node returns a list of all cluster leaders on the next lower layer. The joining node contacts every of those cluster leaders and determines the closest node to itself in terms of RTT. The closest node is then contacted to report back its own cluster mates in its lower layer. The closest node of these reported nodes is then again determined and contacted. This process is continued on all layers from top to down until a cluster in layer 0 is reached. The node then joins that specific cluster. Nodes that are physically close end up being in the same cluster when this mechanism is used.

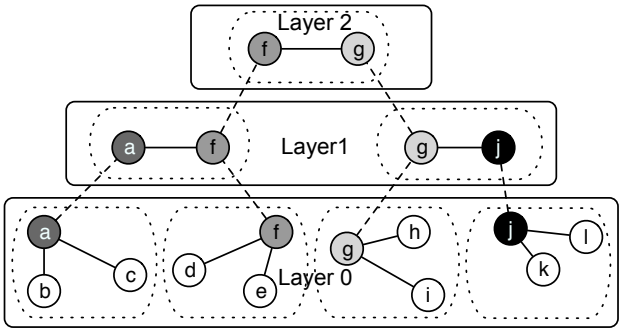


Figure 1.2: NICE Overlay Multicast

When a node leaves the network, it should ideally inform all its cluster mates of its intention before doing so. Then, the cluster leader and the other cluster mates can take appropriate action and update their routing and neighborhood tables. If a node suddenly disappears, then the cluster leader and the cluster mates only recognize the departure of the node by missing heartbeat messages, which have to be sent periodically by all nodes in a cluster.
Maintenance of NICE consists of refinement operations to maintain an optimal tree structure. Refinement operations consist of merging two clusters, splitting clusters, or to determine new cluster leaders. They are only invoked by cluster leaders when invalid or not optimized states are detected in its appropriate clusters.

1.3 QoS for NICE Overlay Multicast

To support QoS for NICE Overlay Multicast, only a minor modification of the NICE protocol has to be performed. Creating QoS aware trees can be achieved by changing the cluster leader determination process. The cluster leaders are now determined by the nodes having the highest QoS class in the cluster (see Fig. 1.3) instead of using the graph theoretic center calculation. This approach is explained in more detail in [6].
To support node to root RTT constraints, the join process has to be modified. Instead of determining the node, which is closest to the joining node on each layer, the joining node also takes the node to root RTT of the reported cluster leaders into account. It then selects one of the nodes, which will provide a node to root RTT below its own node to root RTT constraint. The node that offers the maximum node to root RTT still below its own node to root RTT constraint will be selected from those candidates.

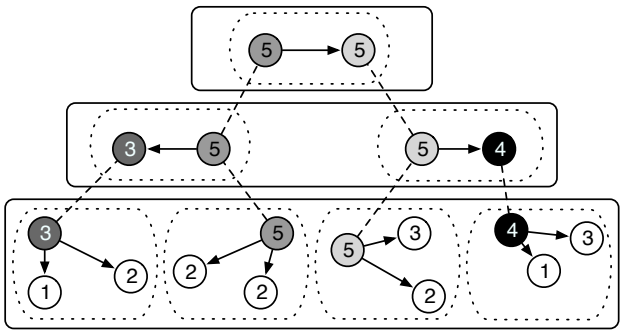
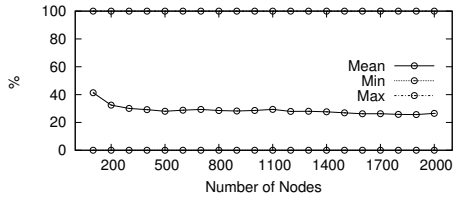


Figure 1.3: NICE Overlay Multicast with QoS Support

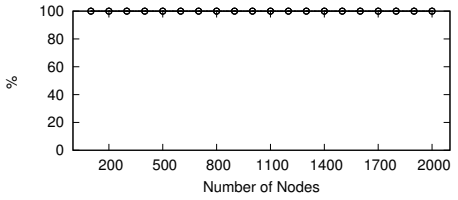
1.4 Evaluation

We evaluated NICE with OM-QoS using the OMNet++ [8] simulator. The basic NICE protocol with enhancements for further reliability using handshakes for split and merge operations, cluster-leader transfers, root transfers and leaving has been implemented for the evaluation.
The following scenarios have been evaluated. Static hard QoS guaranteed by the underlying network; dynamic soft QoS based on measurements offering a best-effort QoS service; and node to root RTT guarantees.
In the hard QoS scenario, the QoS guarantees remain static. Once the QoS reservations have been performed, the underlaying network will guarantee the QoS for the time the node remains in the NICE network. Using soft QoS, which is based on measurements between nodes, the QoS capabilities of a node can change over time. Therefore, links between nodes do not necessarily support the required QoS anymore after a certain time. Then, the node has to find a new parent that supports the initially required QoS on the link between the node and the new parent. Finally, in the node to root RTT scenarios, nodes determine a maximum upper bound constraint for their node to root RTT value. This value is in the range of 25ms–50ms. We then check if this constraint is fulfilled for the normal NICE mode and for the NICE network, which uses a modified join process. We only look at the node to root RTT value directly after the node has joined the NICE network. Finally, we used 13 distance matrices and also varied the arrival, departure, maintance and QoS related values using 3 random seeds. This leads to 780 simulation runs per scenario. We also removed 0.5% of the min. and 0.5% of the max. values in the simulation results of the different runs.

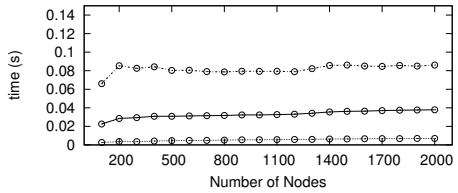
1.5 Normal NICE and QoS Enabled NICE



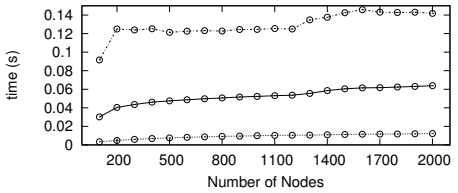
(a) Node to Root Qos, normal NICE



(b) Node to Root QoS, NICE QoS aware



(c) Node to Root RTT, normal NICE



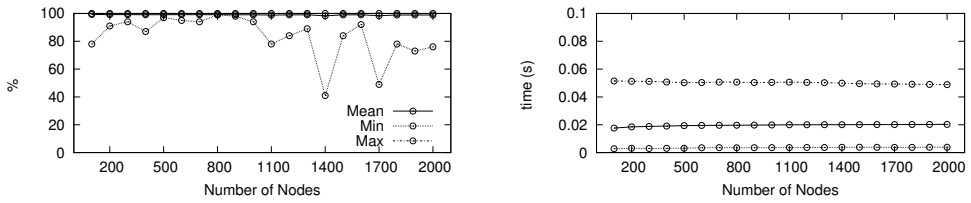
(d) Node to Root RTT, NICE QoS aware

Figure 1.4: Comparing normal NICE with QoS aware NICE

Figure 1.4 compares normal NICE with QoS enabled NICE using static hard QoS guarantees. The value for k was set to 5. The range of the QoS classes was between 0 and 255. We use the normal cluster leader determination process for normal NICE mode and QoS classes were not taken into account. Nevertheless, we analyzed how many paths satisfy the property of monotonically decreasing QoS requirements as described before in a normal NICE network. On average, only 30% of the paths hold the property as shown in Fig. 1.4(a). Using QoS aware NICE as shown in Fig. 1.4(b), 100% of the paths fulfill the QoS requirements. The node to root RTT is presented in Figures 1.4(c) and 1.4(d). Since normal NICE optimizes the RTT between nodes in the same cluster, the average node to root RTT is between 20ms–40ms. In the QoS aware NICE, the average node to root RTT is between 30ms–60ms. Therefore, using QoS adds some overhead in terms of node to root RTT.

1.6 NICE using Soft QoS

The results for dynamic soft QoS are presented in Figure 1.5 with $k = 5$ and 256 QoS classes. While in a NICE network, the QoS guarantees of a node can fail up to 5 times. Figure 1.5(a) shows that using soft QoS does not have a negative impact on the percentage of multicast messages

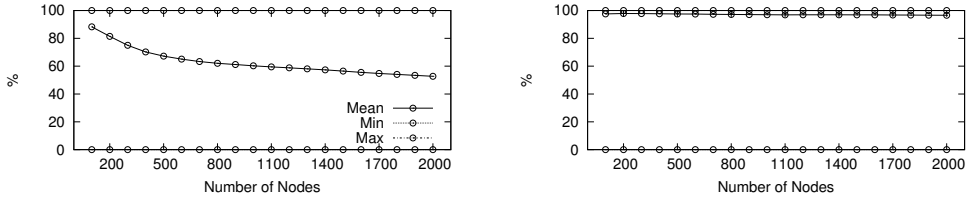


(a) Percentage of Multicast Received (b) Rejoin Duration

Figure 1.5: QoS aware NICE in a Dynamic Soft QoS Environment

received on average. In Fig. 1.5(b), the rejoin duration is presented. The average rejoin duration is around 20ms with a maximum around 50ms.

1.7 NICE with Node to Root RTT Constraints



(a) RTT Constraints Fulfilled, Normal Join (b) RTT Constraints Fulfilled, Modified Join

Figure 1.6: Comparing QoS aware NICE with Normal and Modified Join

In Fig. 1.6, we compare QoS aware NICE with the normal join process and with a modified join process. The modified version takes the node to root RTT constraints into account. Nodes select a random node to root RTT constraint in the range of 25ms–50ms. In Fig. 1.6(a), the percentage of nodes that have their node to root RTT constraints fulfilled using the normal join process is presented. In Fig. 1.6(b), the percentage of nodes having fulfilled those constraints using the modified join process are depicted. Note that the fulfillment of the constraints is checked only directly after a node has joined successfully a NICE network. Using the modified join process enables almost all nodes to have their constraints fulfilled, while for the normal join process, almost half of the nodes do not have their constraints fulfilled for larger number of nodes.

1.8 Conclusion

We presented and evaluated the OM-QoS solution to introduce QoS support for the NICE Overlay Multicast network. The evaluations show that using QoS introduces only a slight overhead in terms of delay compared to normal NICE. The overhead is introduced because of the modified cluster leader determination mechanism. Other aspects of NICE are not significantly altered. We also presented a solution to guarantee certain node to root RTT constraints directly after joining the NICE network. This is achieved using an altered join mechanism.

References

[1] S. Deering, "Host extensions for IP multicasting." RFC1112, 1989.

[2] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, 2007.

[3] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials, IEEE*, 2005.

[4] M. Brogle, D. Milic, and T. Braun, "QoS enabled multicast for structured P2P networks," in *P2PM Workshop at the 4th IEEE Consumer Communications and Networking Conference*, IEEE, January 2007.

[5] M. Brogle, D. Milic, and T. Braun, "Supporting IP multicast streaming using overlay networks," in *QShine: International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, ACM Press, Aug. 2007.

[6] M. Brogle, D. Milic, and T. Braun, "Quality of service for peer-to-peer based networked virtual environments," in *P2P-NVE 2008 Workshop at the 14th IEEE International Conference on Parallel and Distributed Systems*, (Melbourne, Victoria, Australia), IEEE, December 2008.

[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, vol. 32, (New York, NY, USA), ACM, 2002.

[8] Website, "OMNET++, online: http://www.omnetpp.org," 2009.

2 Hypercube-like P2P Overlay for Flexible Routing

Sabina Serbu, University of Neuchâtel
sabina.serbu@unine.ch

2.1 Introduction

In structured peer-to-peer systems, lookup queries follow a certain path given by the routing strategy selection from the routing tables. The path choice can influence the result of the query, the overlay and the underlay network, in terms of accessibility (the information is not found even though it exists), response time (the query takes too long to be answered), or load (the resources at the underlaying level are not equally used). Some of them may get even worse under popular queries, when typically some overlay and underlay links are heavily used, while others are not used at all. In this paper we will deal this kind of demands by providing redundant paths in the overlay structure and allowing to make the right choice between them.

Our solution is based on the HYPEER overlay [1], a logarithmic node degree DHT with a structure that approximates a hypercube. The choice for the hypercube is justified by its support to provide redundant paths. There are other papers that use hypercubes, as [2, 3, 4], however with only limited purpose, such as balancing the request load, dissemination and building routes on demand, respectively.

In this work we refine the HYPEER overlay mostly with regard to routing. Because of the overlay or lookup demands, query lookups have to be restricted in order to fulfill each demand. Thus, we define new routing strategies to comply with the most popular demands, as shortest path length, fault tolerance in case of churn, routing load balancing and low delay path length, while allowing for scalability.

As all other DHTs, HYPEER uses an identifier space where the nodes and the keys obtain IDs in a form of a sequence of binary digits. To route towards a node responsible for the requested key, several intermediate nodes are traversed such that the digits from the source identifier are successively replaced by the digits of the key identifier. The structure can be represented as a ring, but also as a hypercube, where each node has neighbors at exponentially-away distances. This offers the possibility of treating the exponents in any order when routing from source to destina-

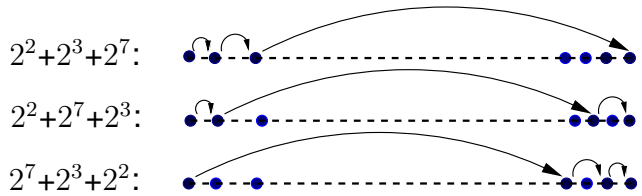


Figure 2.1: Examples of paths on a distance of 140 between the source and the destination nodes.

tion, which tunes the number of remaining redundant paths. Thus, one could think of any lookup or overlay demand that can be solved through routing and define how to make the choice for the next hop. Combinations of the routing strategies can be thought of in order to comply with several demands, however the trade-off is rather obvious.

2.2 Discussion

One of the most known systems with a ring structure is Chord [5]. Each node in Chord has a set of $m = log(N)$ neighbors, where each neighbor is the first node encountered clockwise on the ring after a distance of 2^i, where $i < m$. This means that the identifiers of the neighbors are *at least* 2^i away. When the requests are routed, they follow steps of size $2^i+\varepsilon$, where ε depends on the placement of the nodes on the ring. All links are unidirectional and the routing is clockwise. HYPEER follows the same idea of having m neighbors at exponential distances, but additionally it adds accuracy in the neighbors positions, i.e. it controls the placement of the nodes on the structure, for their identifiers to be at *exactly* 2^i away. This way, the requests would follow hops of size 2^i, that could be taken in any order, which opens way to alternative routing strategies.
As an example, it is straight forward to see that a distance of 140 from node 44 to node 184 can be covered by three hops of 2^2, 2^3 and 2^7 in any order, since addition (their sum is 140) is commutative. The number of possible paths is permutations of the number of hops, which in this case is 3!=6. Three out of the six possible paths are shown in Figure 2.1.

2.2.1 Overlay Structure

Definition We call an *aligned neighbor* a node whose identifier is exactly 2^i away from the current node. An *exact link* is a link that points towards an aligned neighbor.

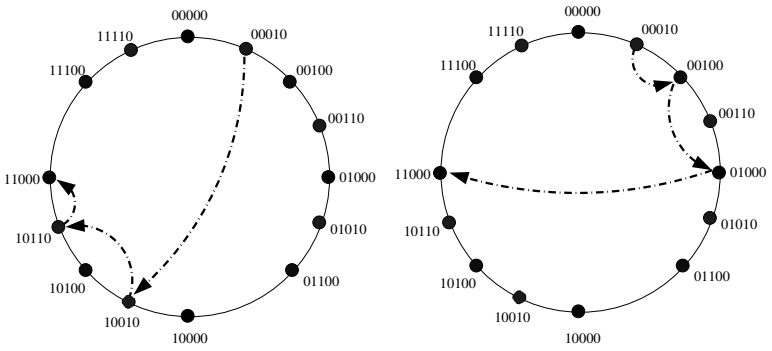
The HYPEER structure spans over a space of 2^m identifiers (where m is the number of bits for each identifier), and can be represented as a classic ring, but, more particularly and which is of more interest for the routing strategies due to its dimensions, as an approximation of a hypercube.

Ideally, all nodes have only aligned neighbors (i.e., all IDs in the identifier space are taken), or all nodes have an aligned neighbor at each of the highest x dimensions (i.e., all IDs of the highest x dimensions in the identifier space are taken), where $x < log(N)$ and N is the number of nodes in the system. However, this cannot happen in the context of peer-to-peer systems, where nodes join and leave all the time, thus the structure cannot be stable. Moreover, all identifiers cannot be taken, otherwise there will be no more room for new node arrivals. This means that by default there are nodes that do not have neighbors on some dimension (i.e., at some 2^i away). In order not to leave those routing table entries empty, we make them point to the first node on the ring after 2^i away, as in Chord. As a consequence, most of the nodes will have neighbors at exactly 2^i away (aligned neighbors), but also, and with a lower probability, neighbors that are at least 2^i away. Our contribution, when comparing with Chord, is to significantly increase the chances for the structure to have exact links in order to adopt new routing strategies.

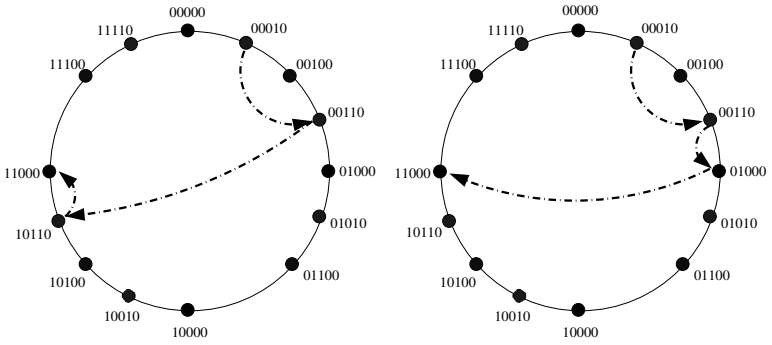
To control the placement of the nodes on the identifier space, we do not rely on a hash function to obtain the identifiers of the nodes, instead we delegate an existing node n_a to assign an identifier for a new node n_b. Node n_a picks its highest dimension that does not contain an aligned neighbor and gives that identifier to n_b. Thus, n_b becomes a new aligned neighbor of n_a.

2.2.2 Overlay Routing

We define four HYPEER-specific routing strategies that send the requests clockwise on the ring structure, which keeps the routing path in $O(logN)$ hops. All make use of the aligned neighbors. A node computes the remaining distance to the destination as a sum of exponential hops (with base 2), and chooses an aligned neighbor that lies at one such exponential distance. These latter neighbors are called eligible aligned neighbors. Thus, when the request is forwarded, the estimated distance to the destination decreases by a 2^i, and the expected number of hops (which is the same for all routing strategies) by one. The only difference between these routing strategies is the choice of i.



(a) SP-HYPEER (b) FT-HYPEER



(c) LB-HYPEER (d) PR-HYPEER

Figure 2.2: Routing path from node 00010 to node 11000 for each routing strategy.

Shortest Path routing. SP-HYPEER uses the furthest eligible aligned neighbor. This is a greedy-like routing strategy, where a request follows long hops in the beginning and then shorter hops towards the end, which under churn is expected to provide the shortest average routing path.

Fault Tolerant routing. FT-HYPEER uses the closest eligible aligned neighbor. A request follows short hops in the beginning and then larger ones towards the end, the identifier space between the current node and the destination being at each hop larger than for SP-HYPEER, which increases the fault tolerance on the request path, as noted in [6].

Load Balancing routing. LB-HYPEER uses a random eligible aligned neighbor, which is expected to balance the *routing load* (the load induced by all requests that are forwarded by a node towards their destinations) on

the outgoing links, and thus, even under Zipf-like requests, to reduce the path convergence.

Proximity routing. PR-HYPEER chooses the eligible aligned neighbor that can be reachable in the lowest delay. It is a local decision which is expected to give on average a lower delay routing path than for the other routing strategies.

An example of lookup under no failures is shown in Figure 2.2. Node 00010 is issuing a lookup request for key 11000, thus the hops to be followed are of sizes 2^1, 2^2 and 2^4. In (a), SP-HYPEER makes hops of decreasing size: 2^4, 2^2 and 2^1. Conversely, in (b), FT-HYPEER makes hops of increasing size. In (c), LB-HYPEER chooses randomly, in this example first the hop of 2^2, then 2^4 and finally 2^1. In (d), PR-HYPEER chooses among the three eligible aligned neighbors 00100, 00110 and 10010 the delay-closest one, which we consider to be 00110; then, at node 00110, among the two eligible neighbors 01000 and 10110 the delay closest one, considered to be 01000; the last hop is the remaining eligible aligned neighbor, which is the destination.

2.3 Conclusion and Outlook

HYPEER is based on the idea of sending the requests on exponential distances, on hops of 2^i with different i. The structure controls the node placement in its identifier space, with neighbors located at exponential distances of 2^i, which makes the structure be thought of also as an approximation of a hypercube. The structure has been shown to be uniform and regular, which is key to deterministically locate redundant paths and route around failures.
We use the awareness of the placement of the nodes in order to achieve shorter average path length, better fault tolerance, better load balancing and lower average path delay. Our preliminary results on the four HYPEER routing strategies have shown that each routing strategy achieves its goal, while maintaining the overlay costs low.
The overlay is scalable, so one could think of any overlay/lookup demand that can be complied with by routing, and tune the choice of the aligned neighbors to be used in the routing strategy. Moreover, the routing strategies could be used alternatively: either each request is sent with a certain routing strategy until it reaches the destination, or the choice is done at each hop. A node that is able to detect unequal routing load on its links or

a high rate of churn, could route the request using either the load balancing or the fault tolerant routing strategies. Thus, the overlay routing may be adaptable to current overlay conditions.
Moreover, replication can be easily applied to HYPEER, using the classical strategy of Plaxton-type overlays [7] of replicating one hop before the node that owns a popular object. More precisely for HYPEER, replication is expected to be highly efficient at the nodes for which that node is an aligned neighbor, since these nodes share most of the traffic for that destination.
HYPEER is thus a scalable, adaptive and extendable hypercube-like overlay that complies with overlay/lookup demands through flexible choice routing.

References

[1] S. Serbu, P. Kropf, and P. Felber, *Improving the Dependability of Prefix-Based Routing in DHTs*, vol. 4803, pp. 206–225. Springer, 2007.

[2] T. Locher, S. Schmid, and R. Watternhofer, "eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System," in *Proceedings of the 6th International Conference on Peer-to-Peer Computing*, pp. 3–11, 2006.

[3] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks," in *First Workshop on Agents and P2P Computing Springer LNCS 2530*, pp. 112–124, 2002.

[4] J. I. Alvarez-Hamelin, A. C. Viana, and M. D. Amorim, "DHT-based Functionalities Using Hypercubes," in *Proceedings of World Computer Congress IFIP WCC*, vol. 212, pp. 157–176, 2006.

[5] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proceedings of ACM SIGCOMM*, pp. 149–160, 2001.

[6] S. Serbu, P. Kropf, and P. Felber, "Fault-Tolerant P2P Networks: How Dependable is Greedy Routing?," in *Workshop on Dependable Application Support in Self-Organising Networks DASSON*, 2007.

[7] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing Nearby Copies of Replicated Objects in a Distributed Environment," in *Proceedings of the 9th annual ACM symposium on Parallel algorithms and architectures SPAA*, (New York, NY, USA), pp. 311–320, ACM, 1997.

3 NetICE9

Dragan Milic, University of Bern
milic@iam.unibe.ch

3.1 Introduction

Embedding RTTs into virtual spaces to obtain an efficient RTT prediction scheme has been a research topic for almost a decade [1, 2, 3, 4, 5, 6, 7, 8]. To date, all methods proposed can be classified as either landmark-based or landmark-less. Landmark-less approaches usually perform a distributed simulation of the physical system. Such a simulation incrementally decreases the total error of the embedding of RTTs in the virtual space and converges toward an optimal solution.
Landmark-less approaches are usually completely distributed without any central components. This makes them very resilient to host failures and network outages. On the downside, landmark-less approaches are usually quite sensitive to violations of assumptions underlying the simulation. For example, one of the most commonly made assumptions is that the triangle inequality holds for the measured RTTs. As shown in [9, 10, 11], however, this is not always the case in the Internet. Such violations lead to undesired effects such as the permanent oscillations of host positions in the simulation; or even worse, the rotation and translation of the whole system within the virtual space. Those effects lead to unstable host positions, making them useful only for a limited time even if there were no changes in the underlying network.
We propose NetICE9, an improvement of the VIVALDI [3] approach.[1]. NetICE9, like VIVALDI, is a distributed simulation of a physical system. This means that each host calculates its position in the virtual space without knowing the whole system. The only information each host has is about its neighbors. Unlike VIVALDI, which simulates a system of hosts connected by springs, NetICE9 is a simulation of a crystallization process. This means that NetICE9 simulates the creation of a crystal structure in a virtual space, where each host corresponds to one atom within this structure. The forces which determine the relative positions of the atoms are proportional to the difference between measured RTTs and distances in the virtual space representing them. In a crystal, those forces are in balance. Essentially, each atom is positioned in such a way that the forces

[1] The name NetICE9 was inspired by a fictional ice isomere named ice-nine described in Kurt Vonnegut's novel *Cat's Cradle*

of repulsion from and attraction to its neighbors are in balance. We also present results of our simulations based on RTTs measured in the Internet [12, 13].

3.2 VIVALDI's Instability

Being fully distributed, it does not require any infrastructure and is very robust against churn. On the other hand, as we will show, VIVALDI tends to be unstable. By instability we mean permanent change of host positions in the virtual space.
The most severe source of the instability is the movement of the whole system, i.e. all hosts participating in the distributed simulation performed by VIVALDI. Since VIVALDI never reaches a stable state, local oscillations never cease. Those small local oscillations of one hosts affects the hosts with it as a neighbor. This change then affects their neighbors and so on. As the final effect, the whole system does not stand still in the virtual space, but instead it translates and rotates.
Movement of the whole system in the virtual space does not change the RTT prediction obtained through VIVALDI much, since the relative positions of the hosts in the virtual space do not change much. But, if we consider how such positions are used, it turns out that the system movement poses a problem. Positions of the hosts are usually used in order to predict RTTs. If the positions are constantly changing, we must obtain a new position of the host, to which we would like to have an RTT estimate. By doing so, we could also just perform an RTT measurement itself, since it involves the same amount of communication. Hence, being able to rely on a host position for a longer time of period is desired.

3.3 Reasons for VIVALDI's Instability

Three host whose RTTs do not satisfy the triangle inequality may be sufficient to render a VIVALDI system unstable. For example, those RTTs could be $2.5, 3$ and 8. Each time one host tries to correct its position relative to another host, it will either decrease the "predicted" value of the RTT 8 or increase one of the other two RTTs (2.5 or 3) to have larger predicted values. Since VIVALDI always optimizes one (randomly chosen) RTT prediction, it never reaches a stable state, since decreasing error in one direction increases the error in the other, which must be then compensated in turn. Figure 3.1 illustrates such a behaviour, that we describe as

“host chasing”. At each simulation step, one hosts tries to reduce its RTT prediction to other host. By doing so, it increases the embedding error towards all other host. This increased embedding error then gets corrected by another host, which in turn increases other embedding errors. The result of this behaviours is the translation of the whole system through the virtual space.

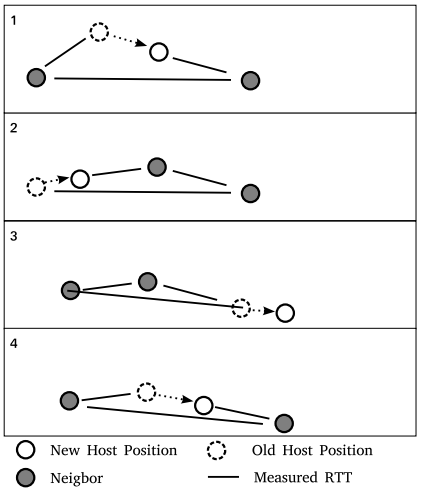


Figure 3.1: An example of “host chasing” in VIVALDI caused by a triangle inequality violation

Another drawback of VIVALDI arises from the fact that VIVALDI does not require the “is a neighbor” relation to be bidirectional. A bidirectional “is a neighbor” relation means, that if host *A* optimizes its position relative to host *B*, then *B* should optimize its position relative to *A*. In practice, each host chooses a fixed number of random neighbor hosts and optimizes its position in the VIVALDI system relative to them. This means, if there is no optimal solution (i.e. some of the properties of the metric space are violated by the measured RTTs), hosts will be oscillating around their (theoretically) optimal position.

Figure 3.2 show what could happend, when the “is a neighbor of” relation is only one way. The host in the middle of the figure tries optimizing its position relative to the neighbors, which at each optimization step leads to a new position of the host, since the neighbors will never move themselves towards the host in the center. If the relation “is a neighbor of” would be bi-directional, the neighbors of the host in the center would also move towards that host, which would reduce oscillation of its position.

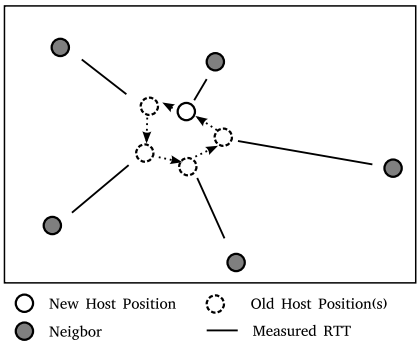


Figure 3.2: An example of oscillations in VIVALDI caused by lack of bi-directional “is a neighbor of” relation

3.4 NetICE9

3.4.1 Crystallization

The basic idea of NetICE9 is to simulate the process of crystallization. Within NetICE9, each host is considered to be an atom that should be embedded into a crystal structure. The crystal structure consists of other atoms positioned within the virtual space in such a way that the forces of repulsion and attraction between them are in balance. We define the force between two atoms to be the difference between the predicted and measured RTT. If this difference is negative, there is a repulsion force between two atoms. Positive difference means the attraction force between two atoms.

Each host determines its own position within the crystal structure. In order to do so, it has a set of neighbors. To each of those neighbors, it periodically measures RTT. At the same time, the host also queries each of the neighbors about its current position in the virtual space and current moving speed (how fast did the neighbor change its position in the virtual space lately). The host uses this information in order to update its own position. Updating the position is done by minimizing the objective function (1). This objective function is very similar to the one used by GNP, with the difference that we are using neighbors as landmarks and that each neighbor is weighted according to its moving speed. The larger the moving speed of the neighbor is, the lower is the weight. The whole process of updating the position of a host is described in Algorithm 1.

The rationale behind using such an algorithm is determining the position of the host using all available information. VIVALDI uses a much simpler approach, where the optimization is done only towards one neighbor. In

the case, where there is no optimal embedding possible, this approach will result in oscillations of the host positions. Those oscillations then have rippling effect on the whole system, as shown in Section 3.2. If we perform the optimization in that way, we propose in Algorithm 1, the host position will remain stable if the positions, velocities and RTTs remain stable. We assume that this would greatly improve both stability and precision (reduce the embedding error) of the system.

$$f_e(\mathcal{C}_\mathcal{H}) \ := \sum_{i=1}^{m} \frac{1}{1 + (\mathcal{V}_{\mathcal{N}_i} \cdot 1000)} \cdot (d(\mathcal{C}_\mathcal{H}, \mathcal{C}_{\mathcal{N}_i}) - \hat{d}_{\mathcal{H}\mathcal{N}_i})^2 \tag{1}$$

Algorithm 1 NetICE9 algorithm for updating host position in a virtual space

Require: N Set of neighbors
Require: T_u Update interval
Require: P_c Current position of the host
Require: V_p Previous EWMA of host's speed
Require: α Dampening factor for EWMA
  for $n \in N$ **do**
    $(RTTS[n], POS[n], SPEED[n]) \leftarrow Query(n)$ {Query a neighbor about its current speed (EWMA), position and measure RTT to it}
  end for
  $P_p \leftarrow P_c$ {Store the current position of the host as the previous position}

  $P_c \leftarrow Minimization(RTT, POS, SPEED, P_p)$ {Determine the new position of the host using function minimization. Parameters for the objective functions are RTTs, current positions and current speed of the neighbors. Use the previous position of the host as the starting point for the function minimization.}
  $delta \leftarrow |P_p - P_c]$ {Calculate the distance between the old and the new position}
  $V \leftarrow delta/T_u$ {Calculate the moving speed of the host}
  $V_p \leftarrow \alpha \cdot V + (1 - \alpha) \cdot V_p$ {Update the EWMA of the host velocity}
  return P_c, V_p

3.4.2 Choice of Neighbors

A crystal structure implies aligning of atoms relative to its direct neighbors. Doing so in order to obtain an RTT embedding is not the best idea.

The authors of VIVALDI [3] showed that using only nearest neighbors for embedding yields in bad results of predicting RTTs of distant neighbors. Instead, they propose using a small portion of distant neighbors, in order to give a host a sense of global position in the network.
We were also assuming that using a bidirectional “is a neighbor of” relation would significantly reduce the rippling effect of local oscillations of hosts to the whole system. The reason for this assumption is that if a bidirectional “is a neighbor of” relation is used, the rippling effect of local oscillations will be reduced, since the effect of local oscillations of two hosts would go both ways and cancel out each other. The problem was that there are not many neighbor selection strategies that have both of those properties. This is why we developed our own neighbor selection strategy called Fisheye [14].
Fisheye is a topology aware overlay network building protocol, which is able to give each host a fisheye-view of an overlay network. A fisheye-view of an overlay network is a choice of c hosts from the overlay network with special properties. Those properties are geographical diversity and fisheye distribution of neighbors.
Geographical diversity means that the choice of neighbors is evenly distributed around the host. Fisheye distribution of neighbors means that the density of chosen neighbors decreases with the distance to the host.
We achieve those properties by performing a distributed gravity force minimization algorithm as described in [14]. Figure 3.3 shows an example, how such a choice of neighbors looks like. An interesting feature of our

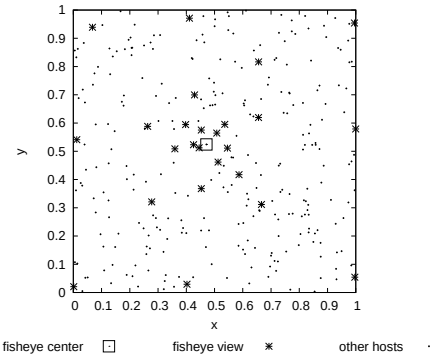


Figure 3.3: Example of fisheye view calculated using gravity force minimization algorithm.

Fisheye approach is, that it does not require embedding of hosts into a virtual space. Instead, it is based only on measured RTTs.
We also developed a version of our overlay network building protocol,

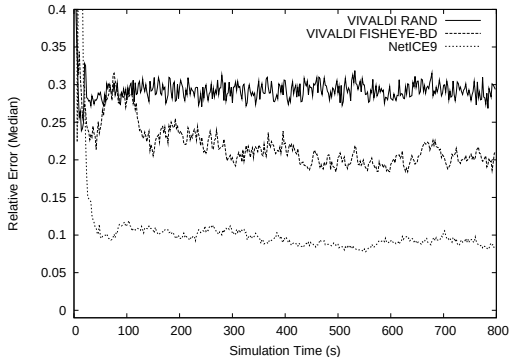
which is able to create a Fisheye overlay network with bidirectional “is a neighbor” relation. This overlay network is the one we are using as a neighbor selection for our NetICE9 approach.

3.5 Evaluation

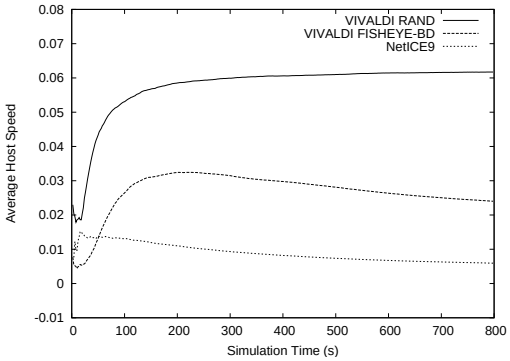
To evaluate the precision and stability of our approach, we compared it to the precision and stability of VIVALDI. To do this, we implemented VIVALDI in an event-based network simulator [15].
The network model of the simulation is the same one used by Dabek et al. in their original evaluation of VIVALDI[3]. Each message within the network is delayed by half the RTT between two hosts. The RTT information for our model was obtained from RTT measurements from the Internet. For input data we hat two data sets. One set (denoted as Planet-Lab) contains a full RTT distance matrix of 217 different hosts obtained from the “all sites ping” experiment [12]. The other data set (denoted as KING) contains a full RTT distance matrix of 462 hosts, which was obtained using the King[13] method.
In each run of the simulation, we started a new instance of a host every 100ms until all hosts were active. This means that all hosts are active by the time the simulation has been running for 21.7 s in the case of the Planet-Lab data or after 46.2 s in the case of the KING data set. For each embedding we used a 5-dimensional Euclidean space as the virtual space. We limited the number of chosen neighbors (c) to maximally 18. We kept each simulation running for 800 s (simulation time).
We compared the median of the relative embedding errors and average host speed in the virtual space for VIVALDI using a random neighbor choice, VIVALDI with a bidirectional fisheye neighbor choice and NetICE9 (a combination of bidirectional fisheye neighbor choice and positioning relative to all neighbors). We performed this comparison for both the Planet-Lab and the KING data set. Results of this comparison can be seen in Figures 3.4(a) - 3.4(d).

3.6 Conclusion

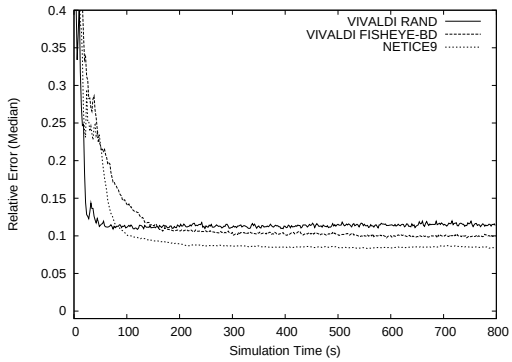
We have identified two major sources of VIVALDI’s instability: One, it does not choose its neighbors symmetrically. Two, it optimizes towards only one neighbor at a time and disregards all information known about the other neighbors. In order to avoid these problems, we proposed a new approach



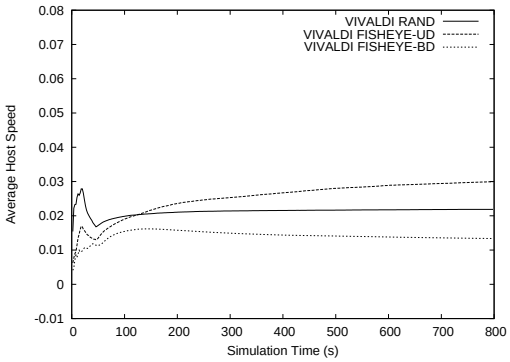
(a) Median of embedding error for NetICE9 compared with VIVALDI using Planet-Lab data.



(b) Average host speed in the virtual space for for NetICE9 compared with VIVALDI using Planet-Lab data.



(c) Median of embedding error for NetICE9 compared with VIVALDI using KING data.



(d) Average host speed in the virtual space for for NetICE9 compared with VIVALDI using KING data.

we named NetICE9. Similar to VIVALDI, NetICE9 is also a fully distributed simulation of a physical system. Unlike VIVALDI, however, NetICE9 simulates the creation of a crystal structure. In this crystal structure, every atom (i.e. every host) positions itself relative to its surrounding atoms (hosts). In order to have a stable system, we chose the surrounding of each atom in such a way that the whole crystal remained stable (i.e. the atoms of the crystal do not move). To achieve this, we propose using such a choice of neighbors that the“is a neighbor of” relation is bidirectional (A is a neighbor of B iff. B is a neighbor of A). Also, motivated by results presented by the authors of VIVALDI, the choice of neighbors should be geographically diverse. This means that for an ideal RTT prediction scheme, a mix of close and remote neighbors should be used. One choice of neighbors fulfilling both of those properties is the fisheye[14] overlay network.

We have evaluated the NetICE9 approach by comparing it to VIVALDI and GNP. For the evaluation we used a simulation based on RTTs measured in the Internet. Our evaluation focused mainly on aspects of stability and precision of RTT embedding. The evaluation showed that NetICE9 outperforms VIVALDI in terms of both precision of RTT embedding and stability.

References

[1] T. S. E. Ng and H. Zhang, "Predicting internet network distance with coordiantes-based approaches," in *IEEE Infocom02*, (New York / USA), June 23-27 2002.

[2] T. S. E. Ng and H. Zhang, "A network positioning system for the internet," in *USENIX 2004*, (Boston MA, USA), pp. 141–154, June 2004.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM '04*, (New York, NY, USA), pp. 15–26, ACM Press, 2004.

[4] M. Costa, M. Castro, A. Rowstron, and P. Key, "Pic: Practical internet coordinates for distance estimation," in *International Conference on Distributed Systems*, (Tokyo, Japan), March 2004.

[5] H. Lim, J. C. Hou, and C.-H. Choi, "Constructing internet coordinate system based on delay measurement," in *Internet Measurement Confgerence 03*, October 2003.

[6] Y. Shavitt and T. Tankel, "Big-bang simulation for embedding network distances in euclidean space," *IEEE/ACM Trans. Netw.*, vol. 12, no. 6, pp. 993–1006, 2004.

[7] Y. Shavitt and T. Tankel, "On the curvature of the internet and its usage for overlay constructi on and distance estimation," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Com puter and Communications Societies*, (Hong Kong / PRC), pp. 374–384, March 7-11 2004.

[8] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Internet Measurement Confgerence 03*, October 2003.

[9] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha, "On suitability of euclidean embedding of internet hosts," in *SIGMETRICS '06/Performance '06:*

Proceedings of the joint international conference on Measurement and modeling of computer systems, (New York, NY, USA), pp. 157–168, ACM, 2006.

[10] B. Zhang, T. S. E. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang, "Measurement based analysis, modeling, and synthesis of the internet delay space," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, (New York, NY, USA), pp. 85–98, ACM, 2006.

[11] E. K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft, "On the accuracy of embeddings for internet coordinate systems," in *In IMC*, 2005.

[12] C. Yoshikawa, "Planetlab all-sites-pings experiment url: http://ping.ececs.uc.edu/ping/," 2006.

[13] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *SIGCOMM Internet Measurement Workshop 2002*, 2002.

[14] D. Milic and T. Braun, "Fisheye: Topology aware choice of peers for overlay networks (submitted for publication)," 2009.

[15] "OMNET++ community site." Available: `http://www.omnetpp.org`, 2009.

4 Energy efficient Multihop Linklayer Protocol

Markus Anwander, University of Bern
anwander@iam.unibe.ch

4.1 Introduction

The Medium Access Control (MAC) layer tries to ensure that no two nodes are interfering with each other's transmissions, and deals with the situation when they do. In wireless sensor networks (WSNs) the MAC layer has an additional aspect. Sensor nodes are generally battery-operated that makes energy consumption is very important. The radio is usually the component that consumes most energy. Therefore main strategy of energy efficient mac protocols to save energy is to keep the periods of active radio transceiver as short as possible and shut down unused radio transceivers immediately. Energy efficient MAC protocols can be mainly categorized as either time division multiple access (TDMA) based or contention based MAC protocols. TDMA based protocols allocate to each node an exclusive time-slot for communication. In these time-slots collision-free media access is guaranteed. Therefore TDMA based protocols need a very accurate synchronization. Contention based protocols usually require a less accurate synchronization, when using a RTS/CTS exchange to enable collision-free media access. When using preamble sampling, no synchronization between the nodes is required.
A new generation of sensor motes, such as the MicaZ, TelosB, and iMote, make use of the Chipcon CC2420 802.15.4 radio. Instead of transmitting a raw bit stream, this type of packetizing radio takes as input the payload of the packet, and the radio module inserts its own preamble, header information and CRC. In comparison to other radio modules the CC2420 is very energy efficient.

4.2 Discussion

X-MAC is a MAC protocol that is working with the IEEE 808.15.4 conform CC2420 radio modul. It uses preamble sampling for media access. Therefore it is possible to use asynchronous listen-sleep cycles enables X-MAC and adaptive listen-and-sleep cycles. To calculate an optimal listen-sleep cycles, X-MAC estimates the traffic load to make a traffic prediction. The

likelihood of k packets arriving over a period of $n \cdot t$ can be modelled as a Bernoulli process of n trials with probability of success $Pd(t)$. The authors of [1] show that $\widehat{Pd(t)} = \frac{k}{n}$ is an optimal instantaneous estimate of the traffic load $Pd(t)$.
A disadvantage of X-MAC is that no hop to hop acknowledgement are provided. This results in a high packet loss in larger multihop WSNs. To minimize the packet loss we try to implement hop to hop reliability with positive acknowledgments. The new protocol is called: Burst enabled Energy efficient Adaptive MAC protocol (BEAM).
First implementation show that traffic prediction has to consider retransmissions. X-MAC just drops packet without any retransmission for reliability. BEAM retransmits packets after biterrors. This leads to a considerable different behavior between X-MAC and BEAM in case of interferences or collisions. Figure 4.1 shows that shorter sleep cycles can result in a lower energy consumption for BEAM.

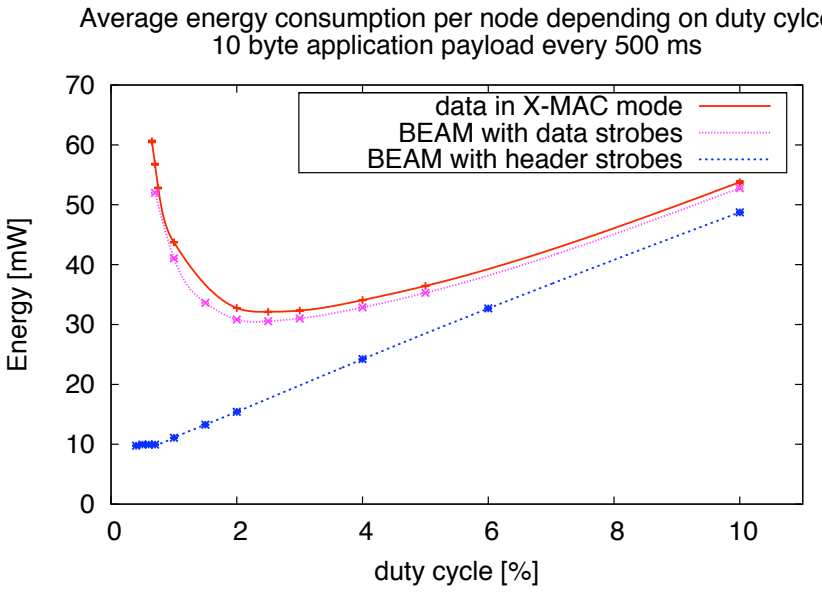


Figure 4.1: Energy consumption with different duty cycles

This is mainly caused by less required header strobes. An other effect is that less header strobes also lead to less interference.

It was not possible to find an optimal traffic prediction including retransmissions by using values like LQI (Link Quality Indication), CORR (CC2420 correlation value), RSSI (Received Signal Strength Indication), error rate or packet size. Main reason is the impossibility of analyzing a corrupt frame and the hidden node problem. To solve that problem we introduce a traffic indicator a. To calculate the traffic indicator every node adds the amount of expected packets for a target node to 2 unused bits of the IEEE 802.15.4 header. Out of the received amount of expected packets the receiver nodes calculates its traffic indicator a. The new traffic prediction for BEAM, enhanced by the traffic indicator, is $a\ \widehat{Pd(t)} = \frac{k \cdot\ a}{n}$.

We found out that a significant part our hidden node problem is caused by the omnet++ radio modell that is designed for IEEE 802.11 radios. To calculate the biterror probability respectively the Packet Reception Rate (PRR) the Signal-to-noise ratio (SNR) is used. A CC2420 radio can receive packets with a much less SNR than a IEEE 802.11 radio can do. We adjust the radio modell of omnet++ with the SNR/PRR values determined in [2].

The current biterror probability is calculated by:

$$1 - \left(0.5 \cdot\ e^{\frac{-32 \cdot\ SNR \cdot\ bandwidth}{bitrate}}\right)^{length}$$

4.3 Conclusion and Outlook

Current results show that BEAM requires less energy in larger multihop scenarios with reliability. In scenarios with low packet flow and low bit errors negative acknowledgements seems to the better choice in view of energy efficiency. In scenarios with higher packet flow and higher bit errors positive acknowledgements seems to the better choice in view of energy efficiency and reliability. Figure 4.2 shows the energy consumption of reliable BEAM und unreliable X-MAC. BEAM has no packet loss, X-MAC over 90%. Four connections are sending every second a packet over 6 hops.

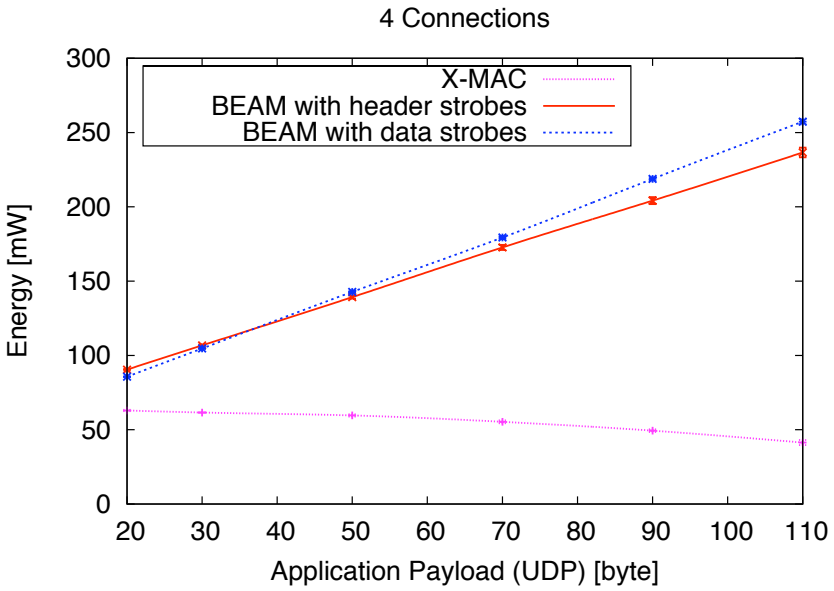


Figure 4.2: Energy consumption of reliable BEAM und unreliable X-MAC

We would like to analyse the behavior of a hop to hop reliability realized with negative acknowledgements. This kind of negative acknowledgements should have considerable advantages to the already implemented end to end reliability with negative acknowledgements. In a hop to hop mode no additional sequence numbers are required and expensive end to end retransmissions can be avoided.
An other idea is to a redundancy layer between the MAC and the network layer. This layer would add or remove dynamically redundant information to the payload depending on the received amount of negative acknowledgements.

References

[1] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 307–320, ACM Press, 2006.

[2] H. Lee, A. Cerpa, and P. Levis, “Improving wireless simulation through noise modeling,” in *IPSN ’07: Proceedings of the 6th international conference on Information processing in sensor networks*, (New York, NY, USA), pp. 21–30, ACM, 2007.

5 Adaptivity of Energy-Efficient MAC Protocols

Philipp Hurni, University of Bern
hurni@iam.unibe.ch

5.1 Introduction

Energy efficiency is a major concern in the design of Wireless Sensor Networks (WSNs) and their communication protocols. As the radio transceiver typically accounts for a major portion of a WSN node's energy consumption, researchers have proposed Energy-Efficient Medium Access (E^2-MAC) protocols that switch the radio transceiver off for a major part of the time. Such protocols typically trade off energy-efficiency versus classical quality of service parameters (e.g. throughput, latency, reliability). Today's E^2-MAC protocols are able to deliver little amounts of data with a low energy footprint, but introduce severe restrictions with respect to throughput and latency. Regrettably, they yet fail to adapt to varying traffic loads and changing requirements of the imposed traffic load.
In my most recent work, I evaluated the energy-throughput and energy-latency tradeoff of today's most prominent E^2-MAC protocols for WSNs. I have motivated the need for more flexible and traffic-adaptive E^2-MAC protocols.

5.2 Discussion

E^2-MAC protocols differ in how nodes organize the access to the shared radio channel. [1] distinguishes three classes of organization *random access*, *slotted access* and *frame-based access*. In *slotted access* protocols, nodes are synchronized to a common sleep/wake pattern. Nodes wake up at designated instants of time to exchange pending traffic. S-MAC [2] is the most prominent protocol of this kind. S-MAC synchronizes the wake-up's of the nodes in so-called synchronization clusters. In each slot, nodes stay awake for an active window of fixed duration. S-MAC applies an RTS-CTS scheme for collision avoidance.
Random access protocols are generally based on contention mechanisms to avoid collisions, and do not rely on synchronized clocks, which makes these protocols rather simple and cheap with respect to the maintenance overhead. Prominent protocols of this class are WiseMAC [3] and B-MAC

[4]. In WiseMAC, nodes periodically wake up to sense the carrier for the presence of a preamble signal, a busy tone that alerts nodes to stay awake for the upcoming frame transmission. A preamble is prepended to each frame to alert the receiving node in its sampling interval. By learning each other's schedules, nodes can minimize the length of the preambles. B-MAC employs a similar wake-up tone scheme to alert receivers, being called low power listening (LPL).
A couple of concepts has yet been applied to reach traffic-adaptive protocol behavior in today's literature on E^2-MAC protocols. In T-MAC [5], an increased traffic-adaptivity of the S-MAC [2] protocol is achieved by prolonging the duty cycles of the nodes when so-called activation events occur. However, simulations show that the adaptivity of the protocol is still very limited. T-MAC shuts down the radio still too aggressively and introduces a high delay for multi-hop transmissions. X-MAC [6] is a recent E^2-MAC protocol based on asynchronous listen-intervals. For each packet, X-MAC sends out a strobe of preambles, in between which the receiver can signalize reception-readiness with a so-called early ack. The authors derive a formula for optimal wake/sleep intervals given traffic at a constant rate and outline a mechanism to let X-MAC adapt the duty cycle and the sleep/wake interval to best accomodate the traffic load in the network.

5.2.1 Simulation Settings

We implemented S-MAC, T-MAC, B-MAC, WiseMAC, X-MAC and the reference protocols IdealMAC and simple energy-unconstrained CSMA in the OMNeT++ Network Simulator [7] using the Mobility Framework (MF) [8], which supports simulations of wireless ad hoc and mobile networks on top of OMNeT++. We applied the current, voltage and transmission rate parameters of the CC1020 [9], a byte-level radio transceiver in the 804-940 MHz ISM frequency band. The CC1020 is used by the MSB430 sensor nodes platform [10], which we intend to use in the near future for prototyping maximally traffic-adaptive E^2-MAC protocols on real sensor hardware. For the *slotted* protocols S-MAC and T-MAC, we assume that the nodes' wake-up intervals are synchronized from the beginning of the experiment (the same assumption is found in many MAC studies, e.g. in [3]). For WiseMAC, we assume that nodes are able to sense transmissions in the channel from stations within their *carrier sensing range* $\sim 2\times$ *transmission range* to implement a cheap collision avoidance. Such a mechanism can be accomplished by most of today's radio transceivers by observing the

onboard RSSI value and setting appropriate thresholds.
In order to allow for a fair comparison, we implemented a packet burst transfer mode for each simulated E^2-MAC protocol, such that nodes can transmit queued packet trains in a burst. Nodes can signalize that they have pending packets to the receiver and continue transmitting packets in a burst, receiving an acknowledgmenet for each frame.

5.2.2 Energy-Throughput and Energy-Latency Tradeoffs

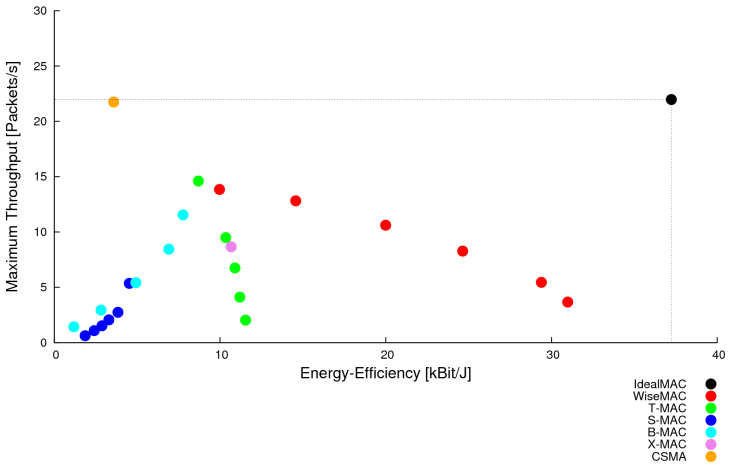


Figure 5.1: Throughput vs. Energy-Efficiency

E^2-MAC protocols typically trade off quality of service versus higher energy-efficiency. Generally, they introduce higher delays and restrain the maximum achievable throughput. In this section, we examine these tradeoffs with the simulated E^2-MAC protocols. By running each protocol with different parameter settings, we thoroughly investigated the behavior of each of the simulated E^2-MAC protocol mechanisms, and not just the behavior of one particular parameter choice. We refer to one parameter tuple for a protocol as a *configuration* hereafter, e.g. one configuration for WiseMAC would be [Basic Interval=200ms, Wake Ratio=1% (2ms)].
Figure 5.1 and 5.2 illustrate the energy-throughput and energy-latency tradeoffs of the simulated E^2-MAC protocols. Each dot represents the results of one particular protocol *configuration* in the simulation experiment. In Figure 5.1, the tradeoff between maximum achieved throughput and energy-efficiency of the simulated E^2-MAC protocols becomes very well visible. CSMA being energy-unconstrained has a very high maximum throughput. However, with CSMA not turning off the transceiver during the low-traffic phases, its energy-efficiency (measured in kBit/J) remains

very low. The IdealMAC protocol, in which a receiver node always *knows* when to switch the transceiver to the receive mode to receive packets, has both a high throughput and a very high energy-efficiency. IdealMAC illustrates where the theoretic lower and upper bounds of the E^2-MAC protocol problems are - it is not possible to reach a higher throughput nor a higher efficiency than IdealMAC. No E^2-MAC protocol will ever get beyond the rectangle that is spanned by IdealMAC in Figure 5.1.
With T-MAC and WiseMAC, the different choices of the frame-length and basic interval parameter values result in dots forming curves that resemble indifference curves. The curves visualize how much maximum achievable throughput the existing protocols need to give up to reach a higher energy-efficiency, when moving from the top leftmost dot towards the lower rightmost dot, and vice-vera. E.g. if WiseMAC is being operated with a very large interval between two wake-up's, the protocol almost reaches the energy-efficiency of IdealMAC, but then only achieves a very limited throughput.
The X-MAC protocol with its wake-cycle adaptation algorithm reaches a mediocre throughput and tolerable delay at a reasonable efficiency, but its performance lags behind that of WiseMAC. The main reason for this is the high per-packet overhead of the preamble strobes. One crucial advantage of X-MAC's strobed preamble mechanism is the possibility to let nodes adapt their wake-sleep cycles to the traffic rate. Nodes with short wake-sleep cycles will respond earlier with an early ACK to the strobed preambles than nodes with a long wake-sleep cycle. Hence the protocol yet offers self-configuration and adaptation capabilities, while with the other protocols, e.g. WiseMAC and T-MAC, the interval between two wake-up's remains constant and does not adapt to the traffic rate.

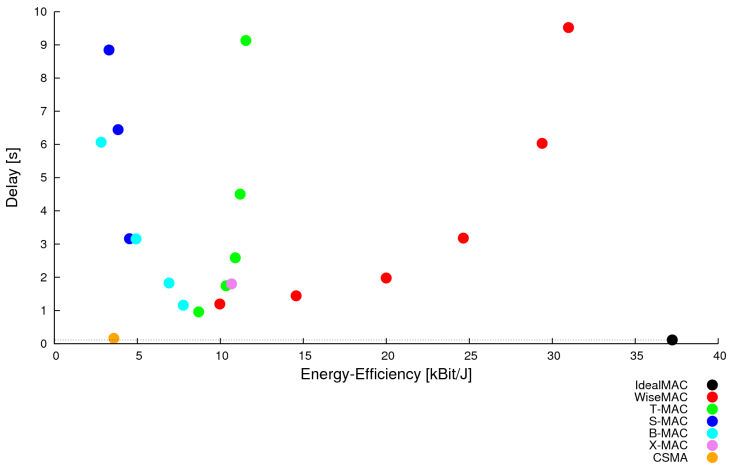


Figure 5.2: Delay vs. Energy-Efficiency

Figure 5.2 similarly depicts the tradeoff between average packet delay and energy-efficiency. One can observe that CSMA exhibits a very low average delay at the cost of a very low energy-efficiency. IdealMAC reaches both, a very low delay at a very high energy-efficiency. IdealMAC again illustrates the lower bounds of the E^2-MAC protocol problem - while it is not possible to reach a higher throughput than IdealMAC, it is neither possible to reach a lower average delay. One can clearly see the energy-latency tradeoff with the different *configurations* of T-MAC and WiseMAC. When increasing the energy-efficiency of the protocol configurations by increasing the interval between two wake-ups, the delay accordingly increases, too. While T-MAC can achieve a lower delay, WiseMAC exhibits a higher energy-efficency.

5.3 Conclusion and Outlook

With this study I have explored the design space of a number of existing E^2-MAC protocols with respect to their ability to react to changing traffic conditions. Today's protocols surely still are from the goal to have an E^2-MAC protocol that truly allocates the radio transceiver in an on-demand manner.
Our intention is to develop an E^2-MAC that is able to achieve a very high efficiency in case of low traffic (as e.g. WiseMAC), but that is capable to adapt its behavior in case of higher traffic, to exploit the entire channel capacity and to achieve a throughput that is similar to that of energy-unconstrained CSMA. Such a behavior would be very advantageous in many event-based WSN application scenarios, and would constitute a real novelty in the design space of E^2-MAC protocols.
We will study mechanisms to let nodes allocate enough resources (by turning/keeping the radio on) to handle increasing traffic load and load peaks in a timely manner. By adaptively switching from one configuration to another at *run-time*, e.g. by shortening the interval in-between two wake-ups when sensing increasing load, nodes reach higher throughput rates in case of high traffic, and switch back to high energy-efficiency at sparse low-rate traffic. Although this may sound straightforward, countless non-trivial problems are certain to arise, as nodes will have to communicate their current configuration among neighboring nodes, which is in turn certain to impact on the energy-efficiency.

References

[1] K. Langendoen, “Medium access control in wirelss sensor networks,” Nova Science Publishers, May 2008. Bookchapter.

[2] W. Ye, J. Heidemann, and D. Estrin, “An Energy Efficient MAC protocol for Wireless Sensor Networks,” IEEE Conference on Computer Communications (INFOCOM), 2002.

[3] A. El-Hoiydi and J.-D. Decotignie, “WiseMAC: An Ultra Low Power MAC Protocol for Multihop Wireless Sensor Networks,” ALGOSENSORS, 2004.

[4] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” ACM Conference on Embedded Networked Sensor Systems (SenSys), 2004.

[5] T. Van Dam and K. Langendoen, “An Adaptive Energy Efficient MAC Protocol for Wireless Sensor Networks (TMAC),” Intl. Conference on Embedded Networked Sensor Systems (SenSys), 2003.

[6] M. Buettner, G. V. Y., E. Anderson, and R. Han, “X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks,” Intl. Conference on Embedded Networked Sensor Systems (SenSys), 2006.

[7] A. Varga, “The omnet++ discrete event simulation system,” European Simulation Multiconference, 2001. http://www.omnetpp.org.

[8] W. Drytkiewicz, S. Sroka, V. Handziski, A. Koepke, and H. Karl, “A mobility framework for omnet++,” 3rd Intl. OMNeT++ Workshop, 2003. http://mobility-fw.sourceforge.net.

[9] Texas Instruments CC1020, “Single-Chip FSK/OOK CMOS RF Transceiver for Narrowband Apps in 402-470 and 804-940 MHz Range,” http://focus.ti.com/docs/prod/folders/print/cc1020.html, last visit March 2008.

[10] M. Baar, E. Koeppe, A. Liers, and J. Schiller, “The scatterweb msb-430 platform for wireless sensor networks,” SICS Contiki Hands-On Workshop, Kista, Sweden, 2007.

6 Scalable Video Streaming to Mobile Devices

Christian Spielvogel, University of Neuchâtel
christian.spielvogel@unine.ch

6.1 Introduction

Video streaming is a technique to transport frames from a sender to a receiver where they are displayed as soon as they are received. The advantages of this frame based — immediate playback are the low start up delay and the fact that no storage space is required by the receiver. Problems are caused by insufficient network or host resources that make real time playback impossible. Effects from insufficient network resources are congestion and packet loss. Effects from insufficient host resources are lost or delayed pictures resulting from the fact that mobile devices are not able to decode and downscale the received content in a timely manner.
A solution to these problems is provided by *Scalable Video Streaming* based on Scalable Video Coding. *Scalable Video Coding* produces multiple dependent streams of the same content, called layers. The advantage of these layers is the possibility of adapting the media characteristics (frame rate, resolution, bit rate) without transcoding. The adaptation can be performed in the temporal, spatial or quality domain. Scalable Video Coding in the *temporal domain* can be applied to support heterogeneous devices with different frame rates. Devices with sufficient resources get the full frame rate (e.g., 30 frames per second), devices with limited resources, like mobile devices, receive a limited number of layers resulting in a lower frame rate (e.g., 15 frames per second).
An application scenario for Scalable Video Coding in the *spatial domain* is the support of devices with different resolutions. For example an HDTV-set with a resolution of 1650x1080 pixels would need all layers to render the video in high quality without using interpolation – for a smart phone it would be sufficient to receive only the base layer with a resolution of 320x480 pixels that can be displayed without discarding pixels.
A scenario for Scalable Video Coding in the *quality domain* is graceful degradation. Graceful degradation is the process of selecting a couple of enhancement layers that are not transmitted in case of insufficient network bandwidth. By dropping descriptions it is possible to adapt the required bandwidth of the stream to the available bandwidth of the network and

avoid random loss.
In the following sections I am going to explain Scalable Video Coding in the temporal, spatial and quality domain in more detail.

6.2 Temporal Scalability

Temporal scaling is used to encode a raw video sequence into multiple layers, each having the same spatial resolution but different frame rates. Decoding only the base layer results in a lower frame rate than that of the original stream. Each enhancement layer stores the missing frames, thus increasing the frame rate with every enhancement layer that is decoded. A block diagram for an encoder producing one base and one enhancement layer can be found in Figure 6.1. According to [1] the base layer and

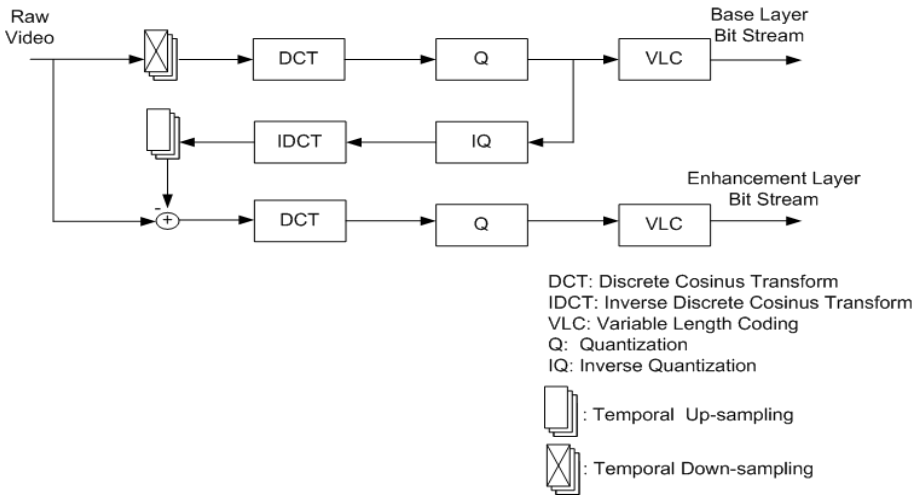


Figure 6.1: Block diagram temporal scalable encoder Scheme

enhancement layer are created in 6 steps. The steps 1 and 6 are identical for the both layers whereas steps 2 to 5 are only required to produce the enhancement layer.

1. The raw video is temporally down-sampled, transformed using the discrete cosine transform (DCT) and quantized. Temporal down-sampling is achieved by skipping frames. For example a down-sampling ratio of 2:1 is achieved by skipping every second frame.

2. In order to produce the enhancement layer each frame is reconstructed by inverse quantization and the inverse discrete cosine transform

3. The input for the enhancement layer is temporally up-sampled to the original frame rate. An exemplary sampling rate of 1:2, can be achieved by duplicating every frame.

4. The difference between the reconstructed and the original frames is calculated, known as the residual.

5. The residual is transformed using the discrete cosine transformation and quantization

6. The quantized coefficients of the (1) base- and (2) enhancement layers are encoded using variable length coding

To decode the base and enhancement layers (1) variable length coding, (2) inverse quantization and (3) inverse cosine transformation have to be applied. The base layer frames are then temporally up-sampled and combined with the difference information stored in the enhancement-layer. A block diagram for a decoder can be found in Figure 6.2.

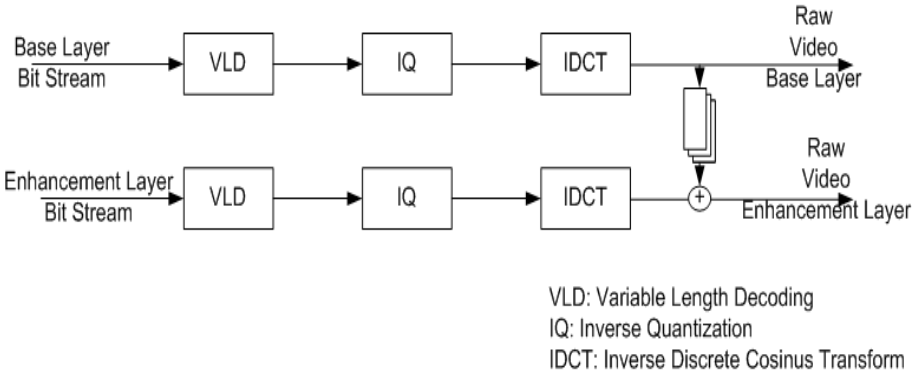


Figure 6.2: Block diagram temporal scalable decoder

6.3 Spatial Scalability

Spatial scaling is used to encode a video sequence into multiple layers having the same frame rate but different spatial resolutions. When only the base layer is decoded the spatial resolution of the resulting video is below the one of the original stream. Decoding the enhancement layer increases the spatial resolution to the original size. A block diagram for the encoder can be found in Figure 6.3. According to [1] base layer and the enhancement layer are created using 6 steps. Once again step 1 and 6

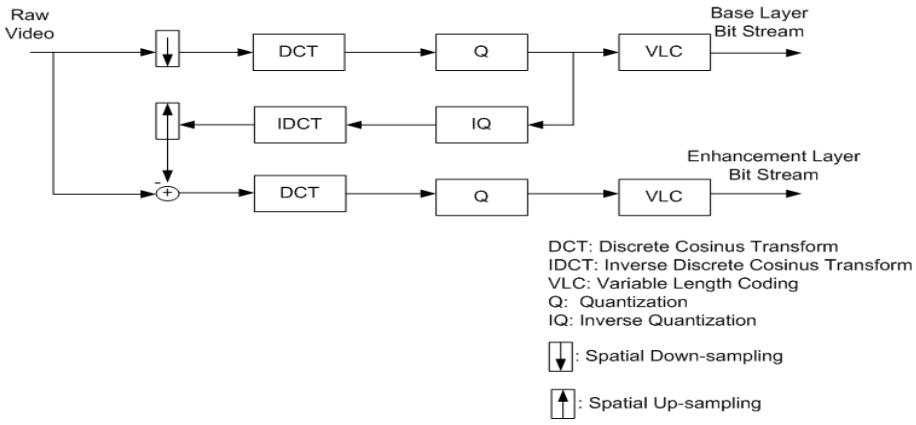


Figure 6.3: Block diagram spatial scalable encoder

are identical for the both layers whereas, steps 2 to 5 are only applied to the enhancement layer.

1. The raw video is spatially down-sampled, transformed using DCT and quantized to get the input for both layers

2. To produce the enhancement layer each frame is reconstructed using inverse quantization and the inverse discrete cosine transform

3. Each frame is spatially up-sampled to the original size using interpolation

4. For the enhancement layer each frame is up-sampled and subtracted from the original image. This difference is known as the residual

5. The residual is transformed using the discrete cosine transformation and quantized

6. The coefficients from the (1) base- and (2) enhancement layers are encoded using variable length coding

In order to decode the (1) base- and (2) enhancement layers variable length coding, inverse quantization and inverse cosine transformation have to be applied. Each base-layer frame is spatially up-sampled and combined with the residual stored in the enhancement-layer. A block diagram of decoding process can be found in Figure 6.4.

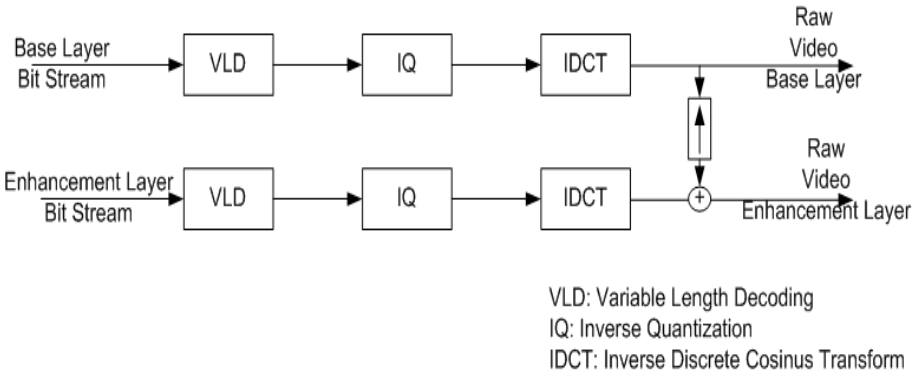


Figure 6.4: Block diagram spatial scalable decoder

6.4 Scalable Coding: Signal-to-Noise-Ratio Scalability

Signal-to-noise ratio (SNR) scaling is used to code a video sequence into multiple layers at the same frame rate and spatial resolution with differing quantization accuracies. The DCT coefficients in the base layer are quantized with a coarse quantizer and the subsequent differences to the original quality are stored in the enhancement layers. A block diagram of an SNR-scalable encoder can be found in Figure 6.5. The encoder produces one

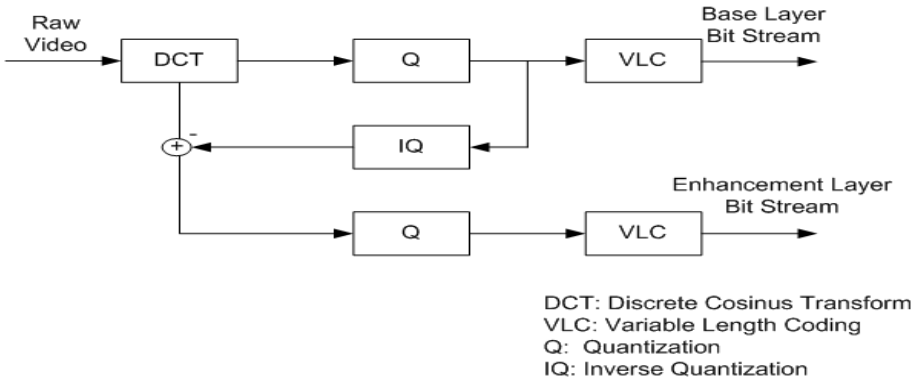


Figure 6.5: Block diagram signal-to-noise scalable encoder

base layer and one enhancement layer. According to [1] the base layer is created by transforming the raw video using the discrete cosine transform, quantizing the coefficients and applying variable length coding. The differences between the highly quantized base layer and the original stream are stored in the enhancement layer using the following steps:

1. The input for producing the enhancement layer is the quantized base layer
2. The DCT coefficients are reconstructed by inverse quantization
3. The reconstructed DCT coefficients are subtracted from the original DCT coefficients. The difference between the DCT coefficients is called residual
4. The residual is quantized by a quantization parameter smaller than the one of the base layer.
5. The quantized bits of each layer are coded using variable length coding

The steps for decoding a SNR scalable video can be found in Figure 6.6. Both the base and the enhancement layers must be decoded using vari-

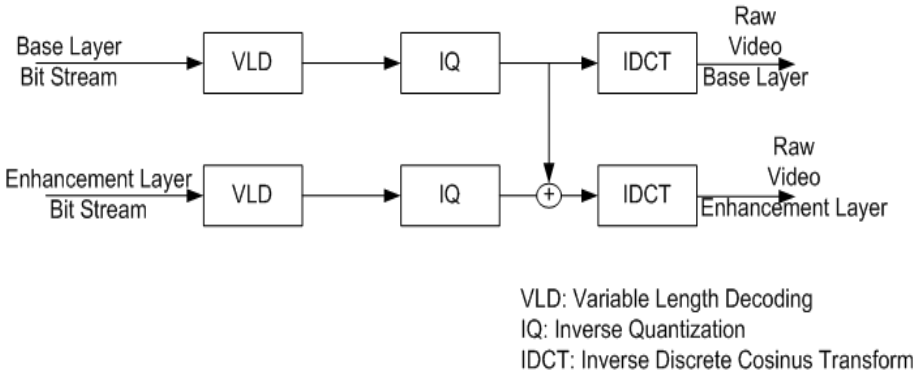


Figure 6.6: Signal-to-Noise Scalable Decoder Scheme

able length coding and inverse quantization. Following this base-layer is modified by the differences stored in the enhancement layer.

References

[1] Dapeng Wu, Yiwei Hou and Ya-Qin Zhang, “Scalable video coding and transport over broadband wirelessnetworks,” *Proceedings of the IEEE*, vol. 89, pp. 6–20, Jan 2001.

7 Challenges in Domain Specific Information Retrieval

Claire Fautsch, University of Neuchâtel
claire.fautsch@unine.ch

7.1 Introduction

Over the last years the amount of electronic data constantly grew and with it the need for efficient information retrieval systems. The data has not only to be made available, but the user also wants to search the available data to satisfy his information needs. A particular case is for example the increasing number of electronically available scientific journals or the emergence of web logs (blogs). In these cases an efficient domain specific information retrieval system is needed. A computer scientist for example will not look for relevant information regarding his domain in a collection of papers issued from a biomedical journal. Consequently the information retrieval system used by the computer scientist can be adapted to show a maximum performance in the domain of computer science, taking account of particularities such as technical language or formulae. The same would be true for any other user looking for information in a particular domain. Hence domain specific information retrieval raises various challenges. In the remain of this technical report, we will present our current work in the field of domain specific information retrieval by means of three different domains.

7.2 Discussion

One of the main goals of domain specific information retrieval (DS IR) is to optimize retrieval systems to efficiently retrieve relevant documents from a collection containing documents from the given domain. Usually the given domain would first be studied to distinguish characteristics and particularities of the domain and then retrieval system would be adapted to consider these.
In DS IR we might for example use a thesaurus or ontologies to extend documents and queries and thus improve retrieval effectiveness. For example while for a computer scientist “Windows” might obviously stand for an operating system, in other domains this word has a complete other meaning. By extending the query with “OS” and “Microsoft” for example,

the query is disambiguated. On the other hand two users might be studying the same object, but using different terms such as for example biologists and medics. In our work, we mainly focus on three domains, namely social science, blogsphere and biomedicine. In the following sections we will give an overview of each of these domains, elaborate the various challenges brought up by them and present our solutions.

7.2.1 Social Science

Our first domain of interest is the social science domain. We conducted several experiments on this domain using a German collection of bibliographic records, the GIRT (German Indexing and Retrieval Test database) corpus in its fourth version (GIRT4-DE) made available through the CLEF[2] evaluation campaign. The collection contains 150,000 records from SOLIS (Social Science Literature) and FORIS (Current research in the field of social science). Each record contains a title and abstract as well as manually added keywords extracted from a controlled vocabulary. Furthermore we used a machine readable version of the German-English Thesaurus for Social Science[3] as well as 125 queries deployed in the domain specific tracks of the CLEF evaluation campaigns from 2004 to 2008. For more information on this collection see [1].
In addition to particularities due to the morphology of the German language rather then the domain (e.g., compound words such as “Computersicherheit” instead of “Sicherheit des Computers”), an interesting feature of this collection is the availability of manually added keywords and of a thesaurus. Due to these facts, the question if manually added keywords really improve retrieval results aroused. In [2] we showed that for the given collection manually added keywords actually improve retrieval results by +13.5% (MAP[4] from 0.2748 to 0.3119). However adding keywords manually is expensive (time and money) if they are not added directly by the author and a human annotator knowledgeable in the domain is needed. We thus wanted to see if eventually automatically added keywords extracted from the thesaurus yield the same improvements. Based on the Jacquard similarity, we extended each document with the k ($k = 50$ for our tests) most adapted keywords from the available thesaurus. As presented

[2] http://www.clef-campaign.org/

[3] http://www.gesis.org/en/services/tools-standards/social-science-thesaurus/

[4] Retrieval measure used in IR to evaluate the quality of the returned results. It varies between 0.0 (no relevant item found) to 1.0 (all relevant items found at the top of the ranked list containing the results)

in [2], we could not improve the mean average precision (MAP), in contrary MAP values decrease by 25.76%. By analyzing the results in detail, we observed that we have an improvement for 22 queries compared to manual expansion.
In a second step, rather then expanding documents we tried to expand the queries. This is mainly interesting if the user searching for information is not knowledgeable in the domain and thus the query formulation lacks domain specific terminology. Our experiments did however not show any improvements for the MAP (+0.31%). By analyzing the results in more detail, we see that we have an improvement for 52 queries and a decrease for 72 queries.
The main reason why automatic document and query expansion do not yield the same improvement as manual expansion is the difficulty to take into account context information for automatic expansion as would do a human annotator.

7.2.2 Blogsphere

During the last years web logs, so called blogs, overwhelmed the World Wide Web. Easy to use publishing tools, free web space as well as a growing community promoted the popularity of blogs. Together with the often subjective content of the posts, the growing amount of blogs constitutes an interesting playground for domain specific information retrieval and opinion minding.
To analyze this second domain of our choice we used the Blogs06 collection made available through the TREC[5] evaluation campaign. Among others, this collection contains 3,215,171 permalink documents crawled between December 2005 and February 2006. A total of 150 queries are available for this collection, deployed in the Blog track during the 2006th to 2008th editions of TREC. For a more detailed description of this corpus see [3].
We pursuit two objectives in the blogsphere, first we wanted to improve factual retrieval and second propose an approach for opinion minding. The aim of factual retrieval is to find blog posts relevant to a given query. The first observations made showed that contrary to ad-hoc retrieval stemming hurts retrieval performance and queries are very short and often proper names or company names. Consequently we proposed to ignore stemming and complete the index by using compound constructions as indexing units. Several test runs showed that both strategies would considerably

[5] http://trec.nist.gov/

improve retrieval, as shown in [4, 5]. Furthermore we used Wikipedia to enlarge queries and thus obtained an average improvement of +2.75% ([5]).
The second objective was to propose a first basic framework for opinion minding. The task is to automatically separate blog posts into opinionated and non-opinionated posts and determine whether an opinionated post contains a positive, negative or mixed opinion. We proposed two approaches based on characteristic vocabulary to assign one of four opinion flags to each document (positive, negative, mixed or neutral). These approaches are still at a very early stage of development, but first results can be found in [5]. A major conclusion is that before applying opinion detection features it is important to have a solid baseline retrieving documents relevant to the given query.

7.2.3 Biomedicine

Last but not least, we investigated domain specific information retrieval on the biomedical domain. Due to its very specific language the biomedical domain provides interesting challenges for domain specific information retrieval. Furthermore the growing availability of electronic versions of biomedical journals claims highly efficient retrieval systems adapted to the domain.
To evaluate information retrieval in this particular domain, we used a collection of 162,259 documents extracted from 49 biomedical journals as well as 36 queries expressing real information needs from biologists and relating to one of 14 possible biological entities (e.g., antibodies or gene), both made available through the TREC evaluation campaign.
As proposed already by others ([6]), one of our suggestions to improve retrieval was to enlarge the queries with orthographic variants since the orthography of several domain specific terms is not always well defined in the biomedical domain (Creutzfeld-Jakob or Creutzfeldt-Jacob, Crohn or Krohn, Nurr77 or Nurr-77). Due to the high density and importance of specific words in biomedical documents, our second suggestion was to give more weight to specific words in the query than to general words (e.g., “D.N.A” is more important than “change”). To fulfill this assignment we used the WordNet[6] thesaurus. The results presented in [7] show that it is important to have good baseline retrieval. These additional information improve results but not on a large scale.

[6] http://wordnet.princeton.edu/

7.3 Conclusion and Outlook

We analyzed three domains and tried to improve information retrieval on these domains. After these experiments a few characteristics of domain specific information retrieval can be pointed out. Each domain has its particularities and retrieval should be subsequently adapted to take account of these. For each domain it is however important to have a solid baseline retrieval system. A good "general" retrieval model will also work in a specific domain. A good method proved to be to first do baseline retrieval and then add a second layer to the system adding domain specific features such as query expansion or spell checking. Furthermore external resources, such as Wikipedia, WordNet or thesauri can help to improve retrieval.
Our future attention will be on one side in eventually improving opinion detection and on the other side in investigating new domains such as for example intellectual property and prior arts search. Additionally we want to present a generic model taking into account domain specific information but usable on all domains without adaptation.

Acknowledgments This research was supported in part by the Swiss NSF under Grant #200021-113273.

References

[1] M. Kluck, "Die girt-testdatenbank als gegenstand informationswissenschaftlicher evaluation," in *ISI* (B. Bekavac, J. Herget, and M. Rittberger, eds.), vol. 42 of *Schriften zur Informationswissenschaft*, pp. 247–268, Hochschulverband fr Informationswissenschaft, 2004.

[2] C. Fautsch and J. Savoy, "Comparison between manually and automatically assigned descriptors based on a German bibliographic collection," in *Proceedings of the 6th International Workshop on Text-based Information Retrieval (TIR 2009)*, 2009.

[3] C. Macdonald and I. Ounis, "The TREC Blogs06 collection : Creating and analysing a blog test collection," *DCS Technical Report Series*, 2006.

[4] C. Fautsch and J. Savoy, "Stratgies de recherche dans la blogosphre," *Document Numrique*, vol. 11, pp. 109–132, 2008.

[5] C. Fautsch and J. Savoy, "UniNE at TREC 2008: Fact and opinion retrieval in the blogsphere," in *In The Seventeenth Text REtrieval Conference Proceedings (TREC 2008)*, 2008.

[6] S. Abdou and J. Savoy, "Searching in Medline: Query expansion and manual indexing evaluation," *Information Processing & Management*, vol. 44, pp. 781–789, 2008.

[7] C. Fautsch and J. Savoy, "Ir-specific searches at trec 2007: Genomics and blog experiments," in *The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings*, 2007.

8 Testing and evaluating Distributed Algorithms

Derin Harmanci, University of Neuchâtel
derin.harmanci@unine.ch

8.1 Introduction

While developping distributed algorithms are inherently difficult, their implementation on a set of distributed machines that use message passing as a communication means is also complicated and time-consuming. Unfortunately, there is no other way than an implementation for testing and evaluation of an distributed algorithm. Although it is possible to implement a distributed algorithm in a simulation environment for a single machine, testing and evaluating an algorithm fully is not possible without an implementation on a testbed with physically distributed machines. Moreover, the execution time of a distributed algorithm on multiple machines is potentially much faster than a single machine simulation environment. In this report, we focus on the challenges of implementing a distributed algorithm on multiple machines and how to cut-down those implementation efforts using the **teDA** framework that has been developped for classroom use.

8.2 Discussion

Having designed a distributed algorithm (based on message passing), we would ideally like to transform its pseudocode into an implemetation in a programming language without much effort. For such a transform, functions that correspond to the initialization of the algorithm and to the actions that should be taken by the algorithm upon receipt of messages should be enough, as depicted in Figure 8.1. Although this transformation seems simple, there are many issues to be handled for realizing it. The remaining of the document discusses the challenges in this tranformation, introduces of the teDA framework, points out how teDA addresses the challenges.

8.2.1 Challenges from pseudocode to implementation

The underlying challenges to implement a pseudocode with a programming language is that apart from transforming the core code that cor-

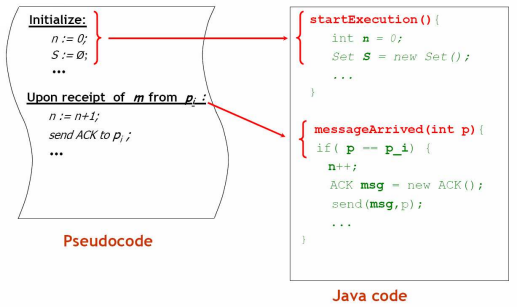


Figure 8.1: Ideal implementation effort in transforming an existing pseudocode to a programming language.

responds to the distributed algorithm, we need to introduce other code addressing several issues to enable a decently working implementation. Those issues can be listed as follows:

- **Messaging subsystem:** This is a fundamental subsystem for the functionality of any distributed system, which should implement the sending and reception of the messages between seperate processes of the distributed algorithm. The major challenge in the design of the messaging system is providing the transparency in the message transport. The distributed algorithm implementation generally does not care where the source and destination of the messages are, it just requests that a message is conveyed from one process to the other through ***send()*** and ***receive()*** primitives. However, the processes can be on the same machine, on seperate machines of the same network or on different networks physically distant from each other. The messaging subsystem should manage those different cases without any visible distinction for the algorithm code.

- **Control and monitoring:** Since we think about a multiple machine implementation it is not practical to think about a debugger. Thus, the ability to monitor events or states of processes is necessary for testing and debugging. Moreover, monitoring is also important for a tested algorithm since external factors that effect algorithm execution can be detected using such mechanism.

- **Injecting failures and unexpected events:** To fully test a distributed algorithm, one needs to test whether it is resilient to failures and whether it reacts correctly upon unexpected events. For this reason, a module that generates such failures and unexpected events in a controlled manner is of great value.

- **Deployment:** Although seems to be a minor issue, deploying a distributed algorithm implementation on multiple (possibly heterogeneous) machines is a very cumbersome and unavoidable task. Again deployment should handle many transparency issues so that the algorithm can just be deployed on selected machines and started by the user in a simple manner.
- **Topology generation:** The virtual topology that is assumed by the algorithm needs to be generated on the existing physical topology. The user of the algorithm should be able to define the mapping between the virtual and physical topologies. Also, it should be possible to generate randomly some known topologies to enable rapid testing and evaluation of the algorithm.

8.2.2 teDA framework

As the previous section suggests, there are many issues that awaits a distributed algorithm designer for distributed environment realization. Meanwhile, an important observation is that all those issues are common to any distributed algorithm (based on message passing), so a tool that provides code modules to be reused and automates some tasks would help overcome those challenges. That is exactly the goal of teDA (**t**est and **e**valuate **D**istributed **A**lgorithms). Although some studies like teDA exist [1], teDA was specifically designed for classroom use.
teDA [2] is designed as a middleware in Java language. The choice of the middleware design and the Java language is no coincidence, because teDA targets many transparency issues over heterogeneous platforms including message passing service between different machines. By design, the middleware expects that each process of the distributed algorithm is a java class which is, generally speaking, named as a *node application*. It also provides an additional specialized class called *master application*. Master application class is not part of the target distributed algorithm but is a control and monitor aid for the user. Several actions performed by the master application are starting the deployment, starting and stoping the algorithm execution and monitoring the other node applications and sending specific messages to insert node failures or message delays.
teDA provides two simple-to-use programming interfaces for transforming pseudocode to Java code. Both interfaces have the same philosophy and we show only the simple interface here just to demonstrate the ease-of-use of the provided interfaces:

```
public interface ApplicationLayer {
public void startExecution();
public void messageArrived(int msgType);
public void incommingBufferOverflow(int msgType);
public void outgoingBufferOverflow(int msgType);
}
```

The first two functions of the interface correspond exactly to the desired transformation shown in Figure 8.1, whereas the last two functions are added to control message buffer overflows if required. With this interface we see that teDA is provides the ideal interface we need for transforming pseudocode to Java code.

8.2.3 Challenges addressed by teDA

Below we introduce the architecture of teDA as well as its different features through an analysis of how it addresses the challenges mentioned in Section 8.2.1.

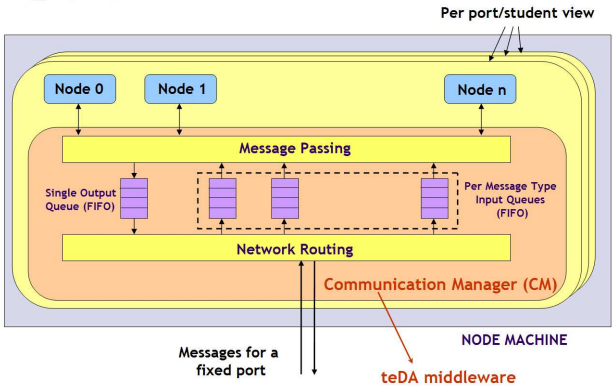


Figure 8.2: Communication Manager of teDA middleware is reponsible for messaging.

Messaging subsystem: The core of the messaging subsystem is the *Communication Manager (CM)* which is depicted in Figure 8.2. CM is the part of teDA which provides the messaging service. It is replicated on each physical machine on which the distributed algorithm runs. The sending of the messages between the processes (*node*s in the figure) of the same physical machine (*node machine* in figure) is directly managed by the CM, while the messages to be sent to distant machines are collected in a FIFO outgoing queue and forwarded to the destination machine using a *network routing* layer. The distant machines receiving the message resolve the target process for the message,

sends a notification to the corresponding process and puts the message to a queue of messages of the same type. For transportation of messages between distant networks teDA chooses to have machines which serve as gateways and communication between distant networks always pass through those machines. teDA installs a *Gateway application* on those gateway machines, This application works like node applications but its task is to forward messages to (or receive messages from) distant networks. Figure 8.3 below shows the distribution of the middleware and how it abstracts the messaging between the processes (node applications) of the distributed algorithm.

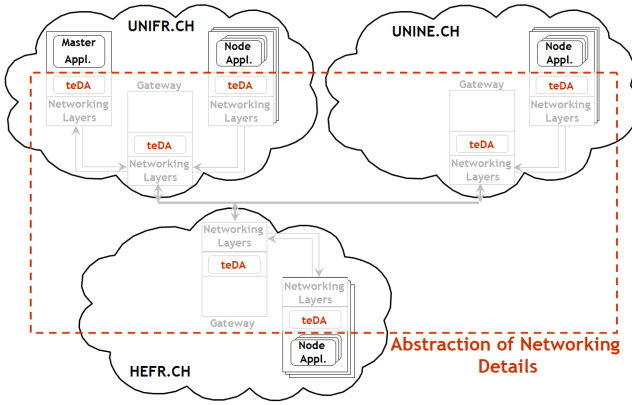


Figure 8.3: Distribution of teDA middleware components over multiple network sites.

Control and monitoring & Injecting failures and unexpected events: teDA mainly uses the messages sent from the master application to node applications to address all those issues. Different type of messages serve for different purposes. For example, the master application can send a ping message to check whether a node application is alive, or it can send pause/resume messages to pause/resume node applications. There is also a message type to set the an additional delay before notification of node application for a message arrival.

Deployment: The deployment of the distributed algorithm is triggered by the master application and it is handled by the gateway applications on gateway machines. The master application packages the distributed application code, teDA middleware code, topology information and application specific configuration information into a single package. It sends this package to the gateway machine responsible for the local network. Then master application unpacks the package in the gateway machine and starts teDA middleware and launces a gateway application on the gateway machine. Once this is done the gateway application reads the topology information and forward the package either on other machines

in the local network or to gateways of the distant networks. The gateway applications of the distant machines are then generated by the gateway application of the local network. All the gateway applications unpack the packages on the machines of their local network and start the node applications that are to be run on each machine. When all node applications are up and running, gateway applications inform master application and the distributed application is ready to run.

Topology generation: Topology generation is done by parsing an input XML file. The XML file structure directly reflects the physical topology where the distributed application will run (the pyhsical machines are identified by their IP addresses). The node applications that are to run on a given machine are listed inside the XML entry for that machine. Also, each node application can have its customized parameters, but those customized parameters needs to be parsed by the corresponding node application.

8.3 Conclusion and Outlook

In this report, an overview of teDA framework has been discussed. We have seen that teDA solves many issues of heterogeneity over network machines and it provides decent platform-independence for implementing distributed algorithms on distrubuted machines. As a tool, teDA provides ready-to-use solutions for cumbersome tasks in implementing distributed algorithms and for this reason it is a good candidate for easy prototyping of distributed algorithms. By the same token, it also serves well as a *distributed algorithms course aid*. With such a tool, students can gain experience in algorithms following either client-server or peer-to-peer paradigms, and evaluate their performances. The permenant/temporary pausing mechanism allows testing fault-tolerance of algorithms. Deferred execution and random termination of processes (node applications) would also provide algorithm development assuming dynamic inclusion and removal of nodes. Last but not the least, it is also possible to evaluate an algorithm's scalability for large number of nodes using teDA.

References

[1] P. Urbán, X. Défago, and A. Schiper, "Neko: A single environment to simulate and prototype distributed algorithms," *Journal of Information Science and Engineering*, vol. 18, pp. 981–997, November 2002.

[2] S. Serbu, M. Schiely, Y. Thiessoz, M. Seifriz, D. Harmanci, M. Sitz, N. Juillerat, B. Hirsbrunner, and P. Kropf, "teDA User Guide," tech. rep., Universities of Neuchâtel and Fribourg, 2009.

9 Transactional memory applied to Application Server

Lucas Charles, University of Neuchâtel
lucas.charles@unine.ch

9.1 Introduction

While modern computers provide multiple cores, programming multi-threaded applications is still a hard and error prone task. In the context of Java EE, concurrent applications usually rely on transactions to ensure the consistency of their data. For instance, a web-shop does not want two consumers to buy the same item, therefore, it needs to ensure that one cannot validate the purchase of an item that might be sold out. To cover this scenatio, Java EE provides transactions through its JTA specification. Which in turn is implemented by Java EE application server such as JBoss. We define a set of operations for which one guarantee that every operations take effect or none of them. If a transaction reaches the end of its execution, it is said to commit, otherwise it is said to abort. A transaction has the following properties (ACID) :

1. Atomicity : Every operations within the transaction will take effect if the transaction commit.

2. Consistency : No data will be left in an intermediate state.

3. Isolation : One cannot read intermediate values.

4. Durability : Changes performed by a committed transaction cannot be revoked.

Usually, a rollback mechanism allows to restart an aborted transaction at the beginning of its execution. A software transactional memory (STM), is a mechanism first introduced by Shavit *et al.* [1] which relies on optimistic assumption to manage concurrency in a multi-threaded application. Nowadays, STM have gained lots of reasearch interests due to the fact that multi-core machines have become ubiquitous. Allowing more concurrency while avoiding the usage of locks and boosting performance if persistence is not required from a web applications leads to the following goals. Providing a concurrent stateful session bean for the Java EE specification, which by running only in memory will avoid the cost of the persistence layer. This work is closely related to the EJB 3.1 specifiaction that will contain a singleton sessions bean to manage concurrent accesses. Also a comparable approch has been taken by Cachopo *et al.* [2], where they use a database

to ensure synchronization among distributed servers, whereas we did not consider ditributed servers. For our experiment, we used LSA-STM developed by Riegel *et al.* [3].

9.2 Discussion

The benchmark is implemented as a crossword game which can be played either by human players or by computers to perform decent simulations. The experiments consist of several java clients, accessing the sever to solve the game. For the needs of the simulation and to ensure the consistency of the results throughout the simulation, clients never solve the problem. They, instead perform random reads and random writes. As stated before, concurrency is managed by using

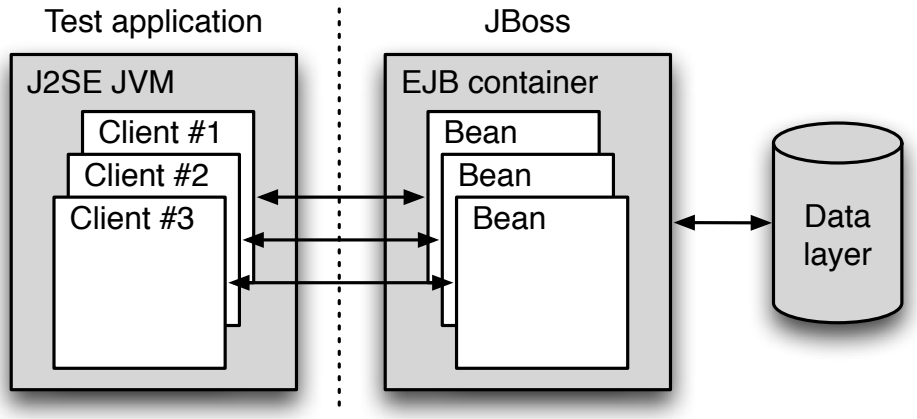


Figure 9.1: Benchmark testing environment.

entity beans, a global lock and a STM. They ensure concurrency and have the following properties :

- Entity Bean :
  - Supported by the hsqldb, bundled with JBoss.
  - Support some degree of concurrency.
  - Will write things temporarily on the disk to ensure perstistence.
- Coarse-grained Locking :
  - Stateful Session Bean, sharing data through static fields.
  - We use a global lock which covers every shared data.
- Optimistic :

- A static field represent the grid, on which we call instance method.
- The grid methods are implemented using calls to the STM, to read or write a word or a letter.

The last mechanism, because of its speculative execution allows the situations presented in Figure 9.2, where conflicts can happen between several concurrent accesses. Thus the STM has to resolve them.

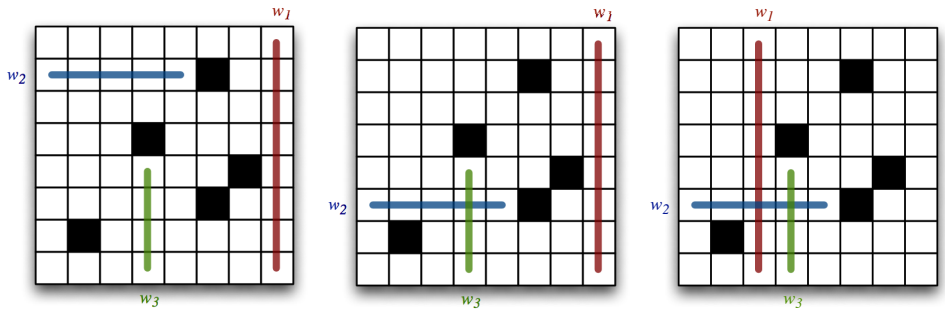


Figure 9.2: Three transactions concurrently entering words w_1, w_2, and w_3 in the crossword application. **Left.** Transactions do not conflict and can all commit. **Center.** Transactions entering words w_1 and w_2 conflict and only one can commit. **Right.** All transactions conflict, but aborting the one entering w_2 allows the two others to commit.

9.3 Results

The results were influenced by three factors; the number of clients accessing the server, the delay used to simulate a payload to avoid serialization due to the network and the size of the grid used for the experiment. In Figure 9.3 we present the results obtained for a grid of size 10x10. One can see that under a big contention, and with few data to manage the entity bean outperforms the STM. One can notice as well that the conflict rate for a delay of zero ms is null, this is due to the serialization effect the network has, as the time to transfer a request is by far longer than the execution of the read and write operations.

Another intersting result shown in Figure 9.4, is how the STM behaves regarding the size of the grid, One can see that the size of the grid does not have a real impact on the STM performance, whereas other techniques suffer when the grid is getting bigger.

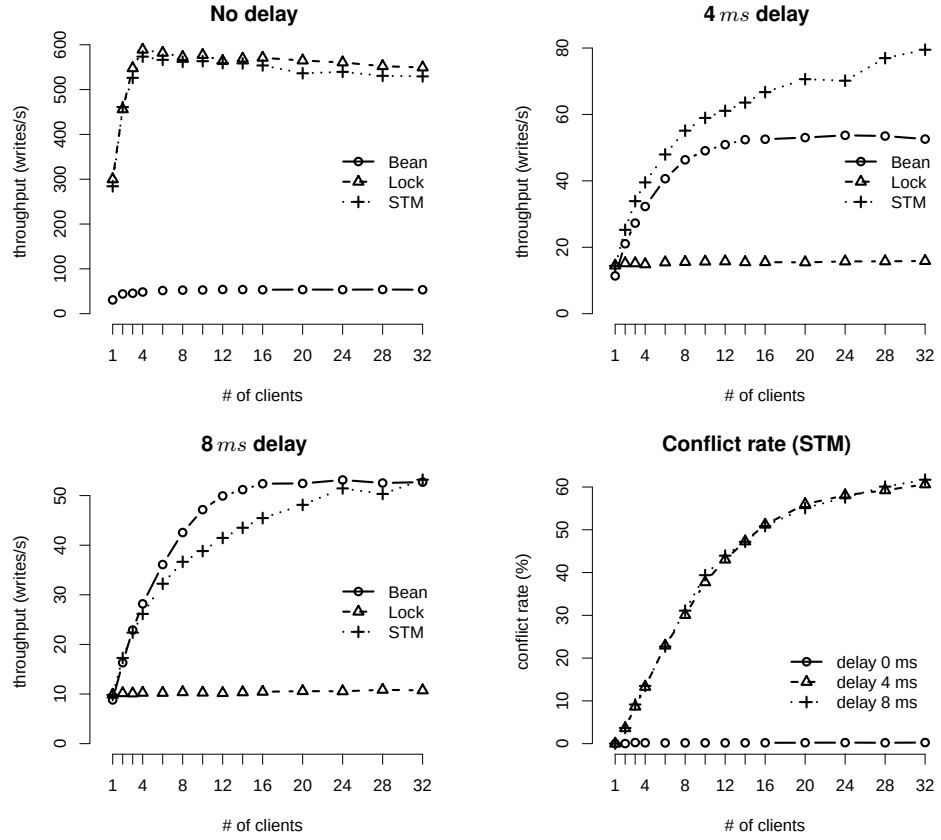


Figure 9.3: Influence of the delay under high contention, with a 10x10 grid. **Top left.** No delay. **Top right.** 4 ms delay between letters. **Bottom left.** 8 ms delay between letters. **Bottom right.** Conflict rate for the STM strategy.

9.4 Conclusion and Outlook

In this experiment we compared three different techniques to manage concurrency in the context of Java EE web applications. We compared entity beans, which are the most common way to achieve synhronization, with a rather unusual global lock scheme and another new method using STM. We implemented a benchmark that allowed us to tune the degree of contention on the server. We ran the experiment to cover a wide range of contention degrees and reported the results. The results obtained show that the STM provides significant improvements over the two other techniques and thus is a good candidate to implement a lightweight concurrent stateful beans, especially when no persistence is required.

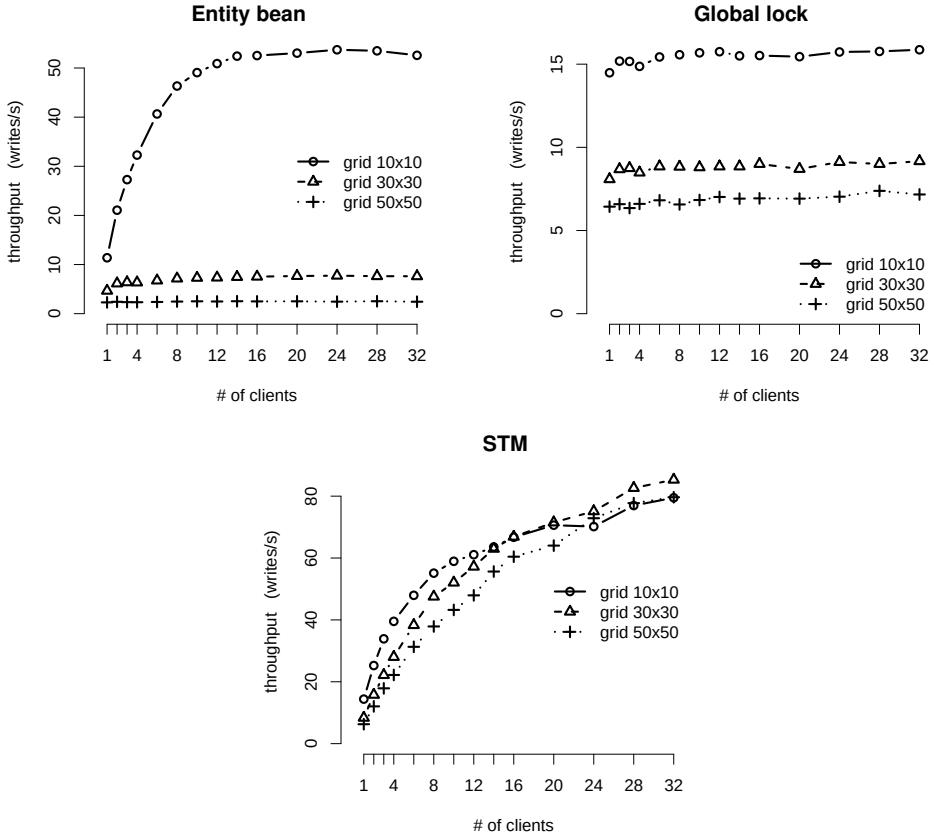


Figure 9.4: Influence of the grid size with 4 ms delay between letters. **Top left.** Entity bean. **Top right.** Global lock. **Bottom.** STM.

Acknowledgment

This work was supported in part by European Union grant FP7-ICT-2007-1 (project VELOX).

References

[1] N. Shavit and D. Touitou, "Software transactional memory," *Distributed Computing*, vol. 10, no. 2, pp. 99–116, 1997.

[2] J. Cachopo and A. Rito-Silva, "Versioned boxes as the basis for memory transactions," in *Proceedings of SCOOL*, 2005.

[3] T. Riegel, P. Felber, and C. Fetzer, "A lazy snapshot algorithm with eager validation," in *In Proceedings of DISC*, September 2006.

10 Event Stream Processing meets Software Transactional Memory

Heiko Sturzrehm, University of Neuchâtel
heiko.sturzrehm@unine.ch

10.1 Introduction

Operations on data streams are done by so called Event Stream Processing (ESP) applications [1]. They typically transform a high amount of low-level events into fewer, more significant high-level ones. This conversion is usually performed by different components that are traversed by the events. Components are able to filter, aggregate or transform the data transported by the events.

Operations on events can be classified as either stateless or stateful. For stateless components, the output stream only depends on the individual input events, e.g., when filtering out events whose values are below a certain threshold. It is easy to improve the throughput of stateless components by just replicating them.

On the other hand, stateful components process events that depend on other events or on some state stored by the component, e.g., when computing the aggregate value of the last n events. In this case multiple instances would have to maintain their common state consistent. Events usually have to be processed in a certain order by a stateful component, either because of interaction between the events or because the state of the whole component relies on an ordered processing of the events. As a consequence, parallel or out-of-order processing is typically not feasible.

A solution to this parallelization problems can be software transactional memory (STM) [2]. It will dynamically detect dependencies between events, if any, and will sequentialize the processing only when necessary (possibly delaying, or aborting and restarting some transactions). Without speculative execution, ESP components would not only have to execute events sequentially one at a time, but may have to *wait idle* when events are not received in the right order. As we shall see, the increased parallelism of merging ESP and STM yields substantial performance benefits.

10.2 Discussion

A first approach to merge ESP and STM was proposed by Brito *et al.* [3] where the TinySTM [4] was altered. A major disadvantage of this work is the strong connection to the LSA algorithm used by TinySTM. In later work (*TM-Stream* [5]), the DSTM2 [6] framework was used. It allows to apply different STM algorithms.

10.2.1 Requirements

Both systems make some assumptions about the way ESP components behave and about the environment.
First, the events receive a logical timestamp as they enter the system. These timestamps need to be unique. To keep that assumption valid throughout the system, each time an event is discarded (e.g., a filter that drops an irrelevant event) a null event is inserted to carry the timestamp through the system.
Second, the algorithms written by the user to process the events should obey certain constraints. All user defined functions should guarantee progress. Furthermore, from within a transaction no external actions can be executed.
Third, we assume that a node has sufficient memory to keep (out-of-order) events in memory until they can be processed and committed.
Finally, we assume that the connections between the components are reliable and events cannot be lost.

10.2.2 Enhanced Component

A stateful component enhanced with support for speculative execution is similar to a regular component from the outside. It supports input and output queues. The main difference is that while events in the input queues may be unsorted, in the output queue, they will be sorted according to their timestamps.
An enhanced ESP component has several threads working in parallel. The number of threads typically depends on the processing capabilities (number of cores) of the system that hosts the component. Each thread can access the input queues and retrieve events to be processed. The manipulation of the event is performed in the context of a transaction, which means that modifications are invisible to other threads until the transaction commits.
The STM-enhanced components use an underlying timebased STM (TinySTM [4] or DSTM2 [6]) that utilizes a shared commit counter to maintain consistent snapshots of memory locations read by transactions without incurring the cost of incremental validation. Commit timestamps are essentially used to linearize transactions and detect whether the content of a memory location can be safely accessed (i.e., is consistent with the transaction execution order). We rely on the use of commit timestamps in our speculative parallelization approach.
Unlike the classical behavior of an STM, transactions cannot complete in any order and threads do not automatically commit their transactions. Instead, transactions have preassigned commit timestamps (determined according to the timestamp of the associated events). A thread checks if a transaction can commit by comparing its timestamp with the current commit counter of the component. If both are equal, the transaction commits and the event can be sent to the output queue. Otherwise, the whole transaction is suspended and inserted in a waiting list. Each time a new event is processed, the list is checked to see if a waiting

transaction can now be committed. It may happen that a transaction in the waiting list is aborted due to a conflict with another transaction. In that case it is restated. As for a regular ESP component, the developer of an enhanced ESP component must provide a function that implements the actual event processing, which shall be protected by a transaction. In addition, s/he may provide other functions which will be executed in different states of the transaction, e.g., after a successful completion of a transaction.

10.2.3 Evalution

In order to evaluate our proposals, we are showing an excerpt of the *TM-Stream* results. Further information can be found in [3] and [5]. First, *TM-Stream* is compared with a common sequential component. The tests were run on an 16-core machine with 4 AMD QuadCore processors at 2.20 GHz running Linux 2.6.25 (64-bit) and Java 1.6_10. The runtime of all experiments was 20 seconds with a warmup period of 1 second. Each point in the following graphs represents the average of three measurements.
The sample application network consists of 5 components as shown in Figure 10.1. Events are generated in the *Input Adapter* and processed in a *Stateless Component*, which can mix the order of the event as they are processed by several parallel threads with a random process time ($0 \leq$ process time $< 100\,ns$). Afterwards, the events are processed in a *Stateful Component*, which receives the events in the sequence given by the *Stateless Component* but have to process them in order. Then, the events are checked in the *Correlator* and, finally, they are collected in the *Output Adapter* that checks their order and produces the statistics.

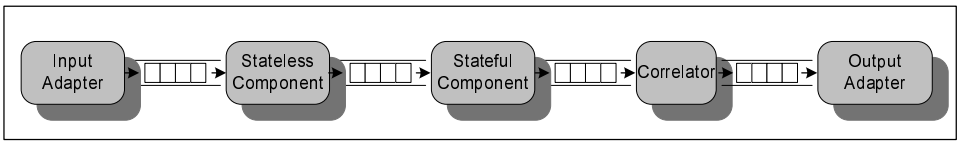
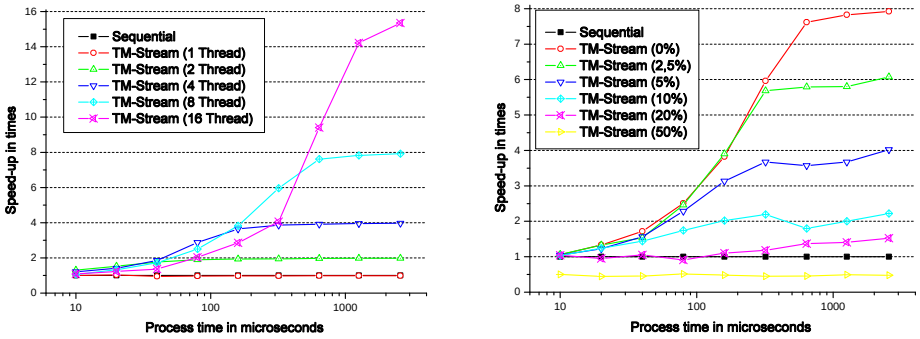


Figure 10.1: Streaming setup

The stateful sequential component first sorts incoming events and then processes them in order. For the *TM-Stream* component, we consider a processing method where every 23rd event has a concurrent access to a local state, i.e., event e_i must be committed before the events $e_{i+(n*23)} \forall n > 0$ can be processed. The prime number 23 was chosen to avoid possible side effects between numbers of threads and the events, since there wouldn't be a speed-up anymore if we have less local states then parallel threads. As already proposed in [3], a predictor is used to delay the processing of those events to minimize aborts. In our case no event is processed if it is more then 22 time units in the future, which makes it a perfect predictor without any abort (except upon conflicting transactions).



(a) Variable Threads and 0% Contention. (b) Variable Contention and 8 Threads.

Figure 10.2: Speed-up of events arriving out-of-order.

For the experiments, the stateless component consists of 8 parallel threads that can produce events out-of-order. The results is shown in Figure 10.2(a). From the figure one can observe that, with increasing processing time per event, the speed-up of the TM-Stream component (with respect to the sequential component) improves. Apparently, each configuration has an optimum that is related to the processing time and the number of parallel working threads. At that point, all threads are fully utilized. This behavior is reflected by the speed-up figure in which the speculative components with more than 1 thread reach a maximum value and then remain flat. Figure 10.2(a) also gives a good indication of the scaling with multiple threads: the speed-up almost reaches the number of threads, which is a very good result.
In another experiment we artificially generated contention, i.e., a certain amount of events are also accessing the local state of their following events. The *TM-Stream* component is using 8 threads in parallel for this setup and the events arrive out-of-order. As a result the abort rate of the STM is increased and events have to be reprocessed, which lowers the speed-up significantly. This becomes apparent in Figure 10.2(b) where the speed-up is reduced dramatically with higher contention. Finally, with more than 20% contention, *TM-Stream* becomes slower than the sequential implementation.

10.3 Conclusion and Outlook

In this technical report we showed a method to improve parallel processing in event stream processing. For speculative processing of events in parallel and delay the commit of the associated transaction until we have successfully processed preceding events, a software transactional memory is used. Both systems can handle ordered as well as unordered incoming events. Within a reason-

able processing time ($\geq 100\,\mu$sec) they scales almost linearly with the number of threads for stateful components. Unfortunately this good scaling can be annihilated through too much contention. Nonetheless it can reduce the working hours for parallelizing ESPs.

Acknowledgment

This research was partly supported by the Swiss National Science Foundation under grant number 5005-67322 (NCCR-MICS).

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widow, "Model and issues in data stream systems," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'02)*, (Madison, USA), pp. 1–16, ACM Press, New York, NY, June 2002.

[2] M. Herlihy and J. E. B. Moss, "Transactional memory: Architectural support for lock-free data structures," in *Proceedings of the Twentieth Annual International Symposium on Computer Architecture*, 1993.

[3] A. Brito, C. Fetzer, H. Sturzrehm, and P. Felber, "Speculative out-of-order event processing with software transactional memory," in *Proceedings of the Second Conference on Distributed Event-Based Systems*, (Rome, Italy), July 2008.

[4] P. Felber, C. Fetzer, and T. Riegel, "Dynamic performance tuning of word-based software transactional memory," in *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2008.

[5] H. Sturzrehm, P. Felber, and C. Fetzer, "Tm-stream: an stm framework for distributed event stream processing," in *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium*, 2009.

[6] M. Herlihy, V. Luchangco, and M. Moir, "A flexible framework for implementing software transactional memory," in *OOPSLA '06: Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, (New York, NY, USA), pp. 253–262, ACM, 2006.

11 Reliable Multicast in IP-based Wireless Sensor Networks

Gerald Wagenknecht, University of Bern
wagen@iam.unibe.ch

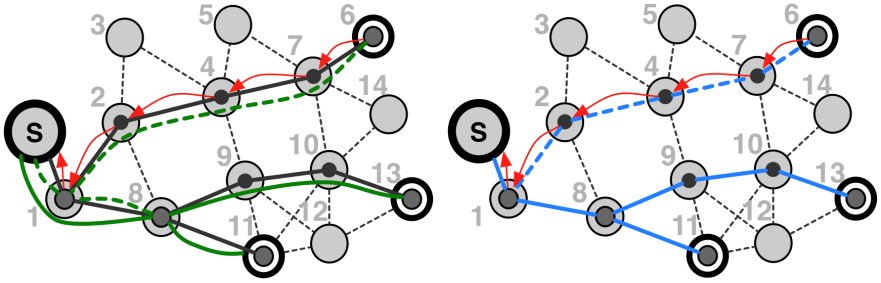
11.1 Introduction

A Wireless Sensor Network (WSN) consists of a number of sensor nodes, which may run different applications for different tasks such as event detection, localization, tracking, and monitoring. Such applications should be configured and updated during the life-time of the sensor nodes and over the network [1]. An update with many unicast connections to the nodes is very inefficient and consumes resources such as bandwidth and energy. Thus it is obvious that multicast communication may benefit the management of WSNs by reducing the number of transmitted packets. To access WSNs via the Internet, a IP-based communication is required [2]. Thus multicast communication should be IP-based as well.
Numerous research has been done about multicast in WSNs. In [3] a multicast protocol called BAM (Branch Aggregation Multicast) is presented, which supports single hop link layer multicast and multi-hop multicast via branch aggregation. VLM^2 (Very Lightweight Mobile Multicast) [4] is a multicast routing protocol for sensor nodes, which provides multicast from a base station to any sensor node, unicast connections from a sensor node to the base station, and mobility. In [5] the authors present an effective all-in-one solution for unicasting, anycasting and multicasting in WSNs. The authors of [6] adapt ADMR (Adaptive Demand-driven Multicast Routing), a multicast protocol for MANETS, on a real wireless sensor node (MICAz). The authors of [7] analyze IP Multicast and show that it is possible to use it in WSNs. Further there are several multicast solutions for WSNs which are based on the geographical position of the sensor nodes in the network [8, 9].

11.2 Designing Multicast in WSNs

Because energy, memory and CPU power are limited in WSNs, existing multicast solutions for wired networks can not be simply ported to WSNs, This implies the following challenges. In wired networks, routers are handling packet replication and forwarding, clients just send and receive simple IP UDP datagrams. Because there are not dedicated routing nodes in WSNs routing functionality for IP Multicast would need to be introduced into each sensor node. Group management is normally concentrated on the routers that communicate with each other to handle multicast trees. The management for groups and multicast trees requires memory and processing power, which is limited on sensor nodes. In general, IP Multicast

is designed to scale on large network groups with multiple receivers and senders. In practical WSNs typically the amount of nodes is rather low. Also the amount of active trees and general management communication should be kept to a minimum. Existing Overlay Multicast [10] solutions (such as Scribe/Pastry, CHORD, Bayeux) are not taking the wireless nature and limited capabilities of sensor nodes into account. Several other issues concerning liveliness, wireless communication and collisions exist. Also reliability for a WSN multicast solution would also be desirable, which are required by code updates and other critical tasks.
Multicasting in WSNs can be designed in two different ways, reliable IP Multicast and Overlay Multicast as described in [11]. Both approaches have source-driven and receiver-driven designs and are centrally managed as well as decentrally. We will distinguish between two node types. Branching nodes have to duplicate packets and store state information about receivers and/or about other branching nodes. Forwarding nodes have less or no information about the multicast state and just forward the multicast data to one neighbor. We will also limit our discussion to core-based trees, where only the dedicated root node will disseminate the data. Our Overlay Multicast protocol implementation is called Sensor Node Overlay Multicast (SNOMC) protocol, and our IP Multicast adoption is called Reliable IP Multicast (REMC) protocol. In this report we will only look at Overlay Multicast / SNOMC (see Figures 11.1(a) and 11.1(b)).



(a) centralized (b) decentralized

Figure 11.1: Overlay Multicast.

11.2.1 Overlay Multicast

For the source-driven scenario we can use a decentralized as well as a centralized approach. In the decentralized approach, the source sends the list of all receivers of the multicast data to its one-hop neighbors. The neighbor nodes check if all receivers in the list can be reached through which next-hop neighbors. If nodes from the list can be reached via different neighbors, the list is split and forwarded to the respective neighbors. The node splitting the list becomes a branching node. UDP connections for the overlay network links are established

between the source, every forwarder, the branching node and the receivers when the corresponding multicast tree becomes active. New nodes are added or removed to/from the multicast tree by sending a join/leave message from the source as described before. Only when a tree is inactive, new nodes can be added. Every node updates its role accordingly to the new members. Upon reactivation of the modified tree, the overlay link connections are opened between all members of the tree. Then the data is transmitted using this overlay links.
In the receiver-driven decentralized approach, receivers send the join message to their neighbor responsible for the default route to the source. Every node on the route to the source becomes a forwarder or a branching node, if it is already a forwarder. Every node prepares its UDP overlay links to the next neighbors accordingly. Receivers that want to leave a group send a leave message towards the source. Forwarders and branching nodes on the path update their status and forward the leave message further.
In the source-driven centralized approach, the source node determines all required branching nodes ahead. Therefore, the source also creates the complete distribution tree that is required for a multicast group. The branching nodes are then notified, process the information and further forward these notifications. Forwarding nodes have no additional information and tasks.
For the centralized receiver-driven approach, the join/leave messages from the receivers are forwarded to the source, which manages the tree as described for the source-driven centralized approach.

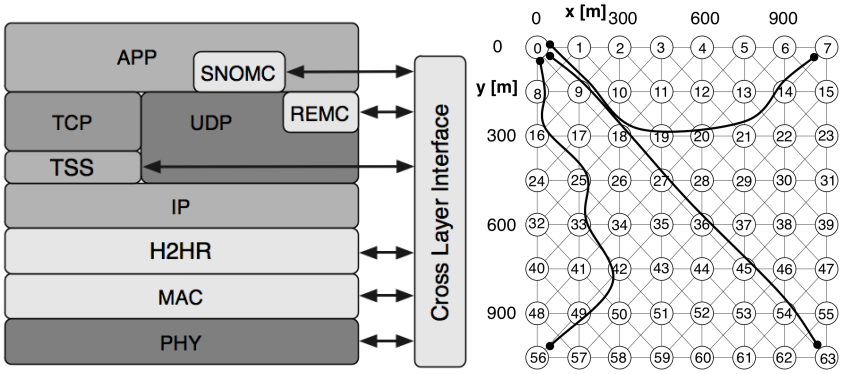
11.2.2 Reliability

To support end-to-end reliability in overlay multicast, we design a simple mechanisms based on intermediate caching on branching nodes, negative ACKs and a closing positive ACK. Every packet has a sequence number. If a packet gets lost, the receiver sends an according NACK message back to the next branching node, which retransmit the lost packet. When all packets successfully arrived the receiver sends a closing positive ACK to the source. On branching nodes the closing positive ACKs are collected and forwarded together to the next branching node or source.

11.2.3 Implementation and Evaluation

We implemented source-driven decentralized mode of SNOMC in the OMNeT++ simulator. SNOMC is embedded in our IP-based protocol stack and shown in Figure 11.2(a). The protocol stack contains a beacon-less 802.15.4 MAC protocol, the Hop-to-Hop Reliability (H2HR) protocol to support reliability for the upper transport protocols, the TCP Support for Sensor Nodes (TSS) protocol to support optimizations for TCP, and our both multicast protocols SNOMC and REMC. To

optimize the performance of the protocols we designed a cross layer interface. All protocols can exchange information across the layers.

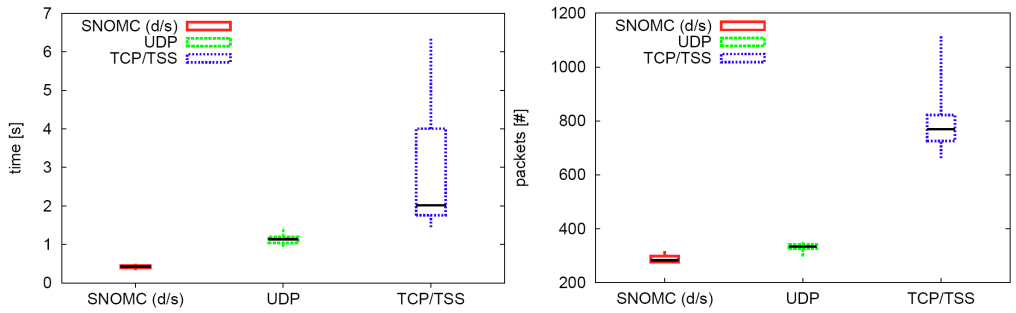


(a) Protocol stack. (b) Simulation scenario

Figure 11.2

We compared source-driven decentralized mode of SNOMC with UDP and TCP/TSS. The scenario is shown in Figure 11.2(b). It is a grid with 64 nodes. Using UDP and TCP we have three connections from node 0 to nodes 7, 56, and 63. Using SNOMC we have the according distribution tree. We transmit 1000 bytes from the source to the receivers. The hop-to-hop reliability is ensured using H2HR. Using UDP we use a simple NACK-based end-to-end reliability mechanism.

The results of the simulation are shown in Figures 11.3. As expected the multicast communication with SNOMC (d/s) is more efficient in terms of time and number of transmitted packet compared with UDP and TCP.



(a) Distribution time. (b) Number of transmitted packets.

Figure 11.3: Comparison of SNMOC (d/s), UDP/E2E, and TCP/TSS.

11.3 Conclusion and Outlook

We introduced different schemes of multicast communication in WSNs and presented Overlay Multicast in more detail. Further we presented a simple end-to-end reliability mechanism based on negative ACKs and a closing postive ACK. We showed that our SNOMC (d/s) protocol reduces the transmission cost significantly compared with UDP and TCP. The main weakness of the current state of the work, that we only consider static routing tables. We will think about to combine the multicast schemes with a kind of on-demand routing scheme and afterwards we can compare both possibilities and see how much it costs to establish the route (and so the tree) on-demand.
The next steps are completing the 4 modes of SNOMC and the developing of REMC (integration into IP Multicast / IGMP). Further, we want to compare our solution with Directed Diffusion and FROMS [12]. And at least we want to implement the protocols on real sensor nodes. We want to integrate them into Contikis uIP / SICSLoWPAN protocol stack.

References

[1] G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, and S. Morgenthaler, “MARWIS: A Management Architecture for Heterogeneous Wireless Sensor Networks,” in *WWIC’08*, (Tampere, Finland), pp. 177–188, May 2008.

[2] M. Anwander, G. Wagenknecht, and T. Braun, “Management of Wireless Sensor Networks using TCP/IP,” in *IWSNE’08*, (Santorini Island, Greece), pp. 1–8, June 2008.

[3] A. Okura, T. Ihara, and A. Miura, “BAM: Branch Aggregation Multicast for Wireless Sensor Networks,” in *MASS’05*, (Washington, DC, USA), November 2005.

[4] A. Sheth, B. Shucker, and R. Han, “VLM2: A Very Lightweight Mobile Multicast System For Wireless Sensor Networks,” in *WCNC’03*, (New Orleans, LA, USA), pp. 1936–1941, March 2003.

[5] R. Flury and R. Wattenhofer, “Routing, Anycast, and Multicast for Mesh and Sensor Networks,” in *INFOCOM’07*, (Anchorage, Alaska, USA), pp. 946–954, March 2007.

[6] B. Chen, K. Muniswamy-Reddy, and M. Welsh, “Ad-Hoc Multicast Routing on Resource-Limited Sensor Nodes,” in *REALMAN’06*, (Florence, Italy), pp. 87–94, May 2006.

[7] J. S. Silva, T. Camilo, P. Pinto, R. Ruivo, A. Rodrigues, F. Gaudncio, and F. Boavida, "Multicast and IP Multicast Support in Wireless Sensor Networks," *Journal of Networks*, vol. 3, no. 3, pp. 19–26, 2008.

[8] D. Koutsonikolas, S. Das, Y. C. Hu, and I. Stojmenovic, "Hierarchical Geographic Multicast Routing for Wireless Sensor Networks," in *SENSORCOMM'07*, (Valencia, Spain), pp. 347–354, October 2007.

[9] J. A. Sanchez, P. M. Ruiz, and I. Stojmenovic, "Energy Efficient Geographic Multicast Routing for Sensor and Actuator Networks," *Computer Communications*, vol. 30, no. 13, pp. 2519–2531, 2007.

[10] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A Survey of Application-Layer Multicast Protocols," *Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.

[11] G. Wagenknecht, M. Anwander, M. Brogle, and T. Braun, "Reliable Multicast in Wireless Sensor Networks," in *FGSN'08*, (Berlin, Germany), pp. 69–72, September 2008.

[12] A. Förster and A. L. Murphy, "FROMS: Feedback Routing for Optimizing Multiple Sinks in WSN with Reinforcement Learning," in *ISSNIP'07*, (Melbourne, Australia), pp. 371–376, December 2007.

12 Cooperative Cognitive Context-aware Composable $(Co)^4$ Virtual Wireless Mesh Networks

Torsten Braun, University of Bern
braun@iam.unibe.ch

12.1 Introduction

Roaming is common in today's cellular wireless networks based on Wireless LAN or GPRS network access. Customers registered at one network operator / provider (e.g., operator A in cf. Fig.12.1) can get network access after registering at another operator (e.g., operator B). Both operators must agree on a roaming agreement in advance, which regulates the terms and conditions of mutual network access by the other partner's customers.

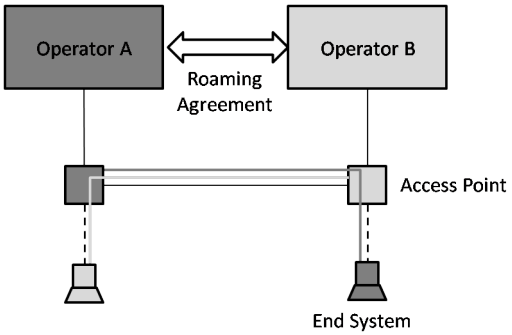


Figure 12.1: Roaming in Wireless Networks

12.2 Wireless Mesh Networks

Wireless mesh networks consist of wireless routers and can extend wireless network coverage in urban areas, provide connectivity in rural or developing areas, provide communication infrastructure in emergency situations, where no network infrastructure exists, interconnect sensor or vehicular networks with infrastructure networks, or provide backbone networks for wireless community networks. In contrast to cellular wireless networks, multiple wireless hops are on a communication path.

12.2.1 Roaming and Cooperation in Wireless Mesh Networks

Also in such a wireless mesh network environment roaming makes much sense. In this case, roaming is implemented by allowing a user registered at operator A to connect to a wireless mesh node operated by operator B (B_1in Fig. 12.2). B_1 then forwards the data to one of the mesh nodes of A (A_1). The data is then forwarded towards the fixed network of B, either via mesh nodes of B only (if at all possible) or via mesh nodes of both A and B.
Roaming requires that operators / providers cooperate with each other. A cooperation agreement must be established among them that defines the rules and procedures of the inter-provider cooperation. For example, it should be defined what kind of foreign customers can get access to an operator's network, which are the services to be offered, and what kind of pricing scheme will be applied. In the case of cooperation, different data flows among the mesh nodes will occur. Management information within a operator's network will be exchanged among the nodes belonging to a single operator, while information specifying the type of inter-domain cooperation will be exchanged among nodes belonging to different operators.

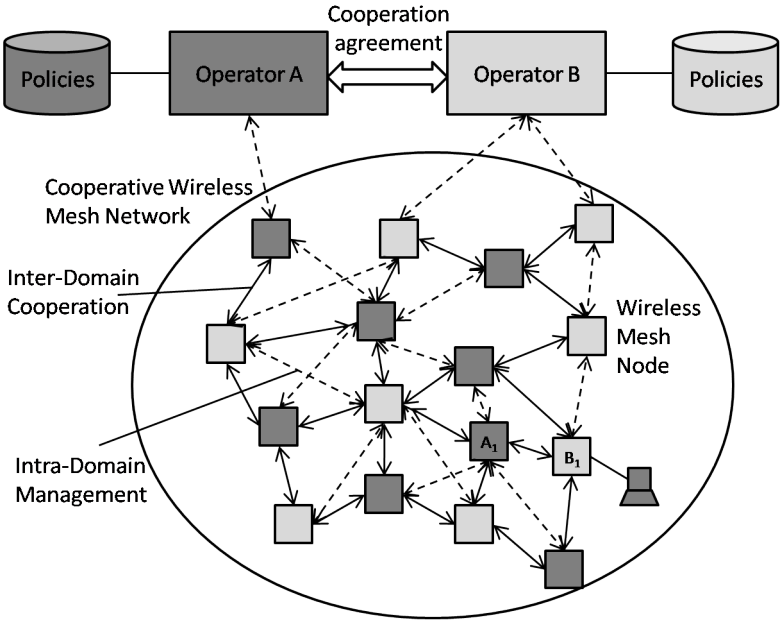


Figure 12.2: Cooperative Wireless Mesh Network

A valid question to be discussed is whether operators should at all be willing to cooperate with other operators. However, there are several benefits for cooperation in wireless mesh networks:

- By sharing wireless mesh nodes among each other, the network coverage for all cooperating operators can be increased.
- Sharing resources such as wireless mesh nodes and frequencies can reduce the equipment and operation costs of a wireless mesh network.
- Interferences can be reduced by reducing the transmission power required to reach a close node of another operator instead of a far node of its own, cf. Figure 12.3.
- In the same scenario, communication with far nodes typically requires usage of lower bit rates than in case of close nodes. Therefore, the communication lasts longer and the frequencies are occupied for a longer time. Communication with close nodes of another operator is more efficient in terms of radio resource usage.

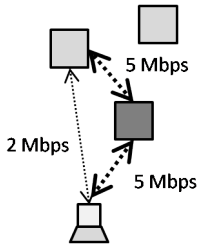


Figure 12.3: Cooperation in Wireless Mesh Networks

12.2.2 Virtual Wireless Mesh Networks

Cooperation in wireless mesh networks can be implemented by virtualization techniques. A mesh nodes belongs to a certain operator, but different mesh routing entities can be run on such a physical mesh node. This may result in multiple different virtual mesh nodes, one for each operator to be supported, cf. Fig. 12.4. Since wireless mesh nodes are expected to run Linux based operating systems, virtual machines can be used for implementing multiple virtual mesh nodes on a single physical node. To support virtualization on lower layers (MAC, physical), different frequencies can be used (frequency multiplexing). Alternatively, time or code multiplexing could be applied.

12.3 Cognitive Wireless Mesh Networks

Wireless mesh nodes should be managed in an autonomic way in order to allow automatic management and to limit management costs. However, wireless mesh

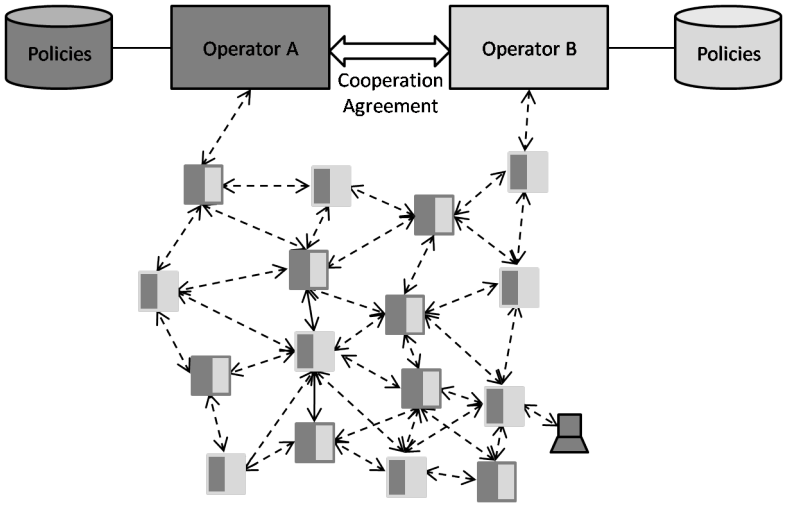


Figure 12.4: Virtual Wireless Mesh Networks

nodes are not always physically accessible, e.g., when placed on roofs that are difficult to access or in areas, where land owners do only allow infrequent physical access. Disruptions and failures must be expected also due to the operation of wireless mesh nodes in rough environments, e.g., in alpine environments with low temperatures and strong winds. Therefore, automatic recovery from disruptions or failures is required. Moreover, configuration inconsistencies must be detected and repaired. We propose a cognitive and context-aware management approach to select the most appropriate parameters and to optimize resource usage based on learned experiences considering context information, e.g., location information or environmental data. The final goal of such a cognitive envronment is to provide self-managing, self-optimizing, and self-healing wireless mesh networks.

12.3.1 Cognitive Networks

A cognitive network is a network with a cognitive process that can perceive current network conditions, and then plan, decide, and act on those conditions [1]. Figure 12.5 the OODA (observe, orient, decide, act) process: The network and physical environment is observed, relevant observations are extracted and based on previously learned experiences a decision is taken. The decision results in several actions such as reconfiguration of network elements. The learning process stores the decisions, the previous network / environment state, the decisions taken, and the resulting network / environment state. Thus, we can build some experience, which actions under which conditions are successful or not. Learning can be assisted by artificial intelligence and machine-learning techniques. Future decisions should take end-to-end goals such as Quality-of-Experience (QoE) monitored on application layers into account.

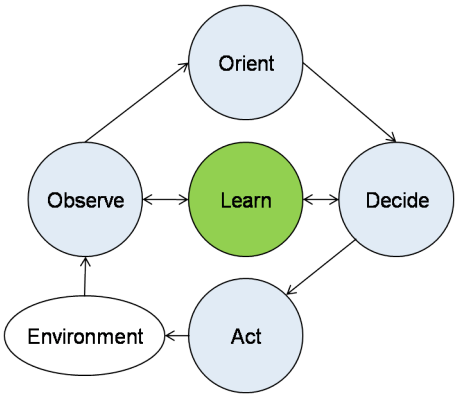


Figure 12.5: OODA Principle in Cognitive Networks

12.3.2 Cognitive Mesh Node Architecture

Figure 12.6 shows a possible cognitive wireless mesh node architecture. The node communicates via external interfaces to other components such as the operator to which it belongs in order to exchange configuration and monitoring information, peer mesh nodes in order to exchange measurement or cooperation information, network sensors in order to retrieve network status information and measurements, as well as other physical sensors providing external context information such as humidity or location information.

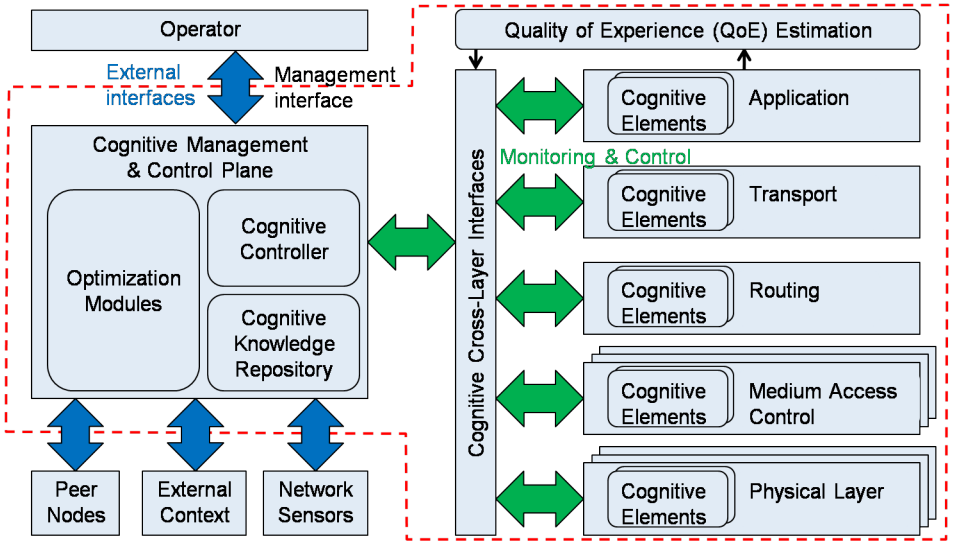


Figure 12.6: Cognitive Mesh Node Architecture

The internal node architecture is based on a *cross-layer architecture*. We assume that we have a layered design with traditional protocols such as physical layers,

MAC protocols, routing protocols, transport protocols, and application layer protocols, where the latter two are only needed for end systems supporting end-user applications. The individual protocol elements consist of functions required by the protocol to be implemented, but also include cognitive elements that allow to decide about the parameters and operation modes to be configured for the respective protocol. Each wireless mesh node has a Cognitive Management & Control Plane, which interacts with the cross-layer protocol stack to set parameters and modes of all protocol layers.
The main part of this Cognitive Management & Control Plane is the *Cognitive Controller* as the cognitive decision making unit. It performs reasoning, triggers actions, and employs learning strategies to optimize system and network operation over time based on historical information. It collaborates with optimization modules and selects the optimization mechanisms depending on the type of the problem and the policies applied. The Cognitive Controller makes its decisions based on monitoring information retrieved from the protocol stack through the cross-layer interface, external context data available through, e.g., sensorial information, and information based on the historical data stored in the Cognitive Knowledge Repository.
The *Cognitive Knowledge Repository*, which has been insipred by [2], assists the Cognitive Controller by storing information elements such as protocol configurations and parameters, resulting performance as well as QoE parameters that have been measured by a QoE measurement entity linked to the application layers. The Cognitive Knowledge Repository holds information gathered from external context sources (e.g. external spectrum sensors) and peers. It is in charge of exchanging information with peers to facilitate distributed optimizations. This is necessary to be able to optimize performance parameters across the whole network.
For scalability and robustness, we explore Peer-to-Peer concepts for exchanging such knowledge information with peer nodes. The set of Cognitive Knowledge Repositories of all nodes forms a distributed knowledge plane that holds network wide information. In the same way, management functions located in the Cognitive Management & Control Plane can access the information at the Cognitive Knowledge Repository and decide based on that about management operations to be performed. By coordinating decisions among peers, distributed cross-layer optimization will be possible. While the cross-layer approaches in [3] are limited to single nodes, we allow cross-layer information exchange between nodes.

12.4 Conclusion and Outlook

We propose the concept of cooperative wireless mesh networks in order to improve the efficiency and operating costs of a wireless mesh network infrastructure. To automate network management, self-* properties shall be achieved by a cognitive wireless mesh network architecture. Specific learning algorithms inspired

by phenomena observed in nature must be developed.

References

[1] R. W. Thomas, D. H. Friend, L. A. Dasilva, and A. B. Mackenzie, "Cognitive networks: adaptation and learning to achieve end-to-end performance objectives," *Communications Magazine, IEEE*, vol. 44, pp. 51–57, Dec. 2006.

[2] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 3–10, ACM, 2003.

[3] V. Srivastava and M. Motani, "Cross-layer design: a survey and the road ahead," *Communications Magazine, IEEE*, vol. 43, pp. 112–119, Dec. 2005.

13 Supporting Wireless Mesh Networks during their Life Cycle

Thomas Staub, University of Bern
staub@iam.unibe.ch

13.1 Introduction

Wireless mesh networks (WMN) are evolving to an important access technology for wireless broadband services. They provide a cost efficient way to interconnect isolated networks as well as to enhance wireless network coverage. WMNs usually consist of static mesh routers and mobile or static mesh clients. Both support multi-hop communication and may act as routers. WMNs offer a robust and redundant communication infrastructure. As in every other network, a WMN and services running on top of it traverse a life cycle, which is split into the following four parts (see Figure 13.1):

Development: In order to provide new services to the customer, the network and the protocols have to be specified, implemented, and evaluated. Usually, these development steps are performed in a network simulator. Afterwards, the protocols and services have to implemented on the target platform.

Testing: During the testing phase, the services and network are tested on the target platforms. Therefore the protocol and service specifications as well as the simulation code has to be migrated and then extensively tested on a real test bed (pre-deployment testing).

Deployment: After the pre-deployment testing the network can be roll out and deployed. The locations of the network nodes have to be specified, the software has to be deployed on the target nodes, and the network configuration has to be setup and distributed.

Maintenance: As soon as the network has been deployed, the maintenance phase starts. During maintenance software updates have to be applied to the network nodes operating system to fix software bugs and to include new functionality. Moreover, the network has to be reconfigured to meet the changing requirements of the users. New requirements may further lead to new development phases.

Figure 13.1: WMN life cycle with four phases

13.2 Challenges

In all phases of the WMN life cycle, challenges for the network engineers occur. In development, it is mainly the accuracy of the simulation model which cannot cover all environmental influence. In the testing phase, the migration to the target platform and the limitation (scale, management) of the test bed cause problems. Afterwards, in the deployment phase, the challenges are more bureaucratic (node locations, regulations) or depend on the weather conditions. Whereas in the maintenance phase, erroneous software or configuration updates may require costly on-site repairs, which are further complicated by physical access restrictions.
The challenges during the development are due to the accuracy of the simulation model. Results of the simulation are influenced by the selection of the wireless propagation model. The propagation model may be to simple to accurately model the real environment. In addition, the influence of the operating system and the drivers are not covered in the network simulations. This includes cross-layer interactions, which are implemented more easily in the network simulation.
In the testing phase the migration to the real hardware is difficult. The programmer has to consider the limitations of the target platforms. The embedded devices are usually restricted in RAM, storage, and CPU power. They further require an operating system and tools, which are especially tailored for them. Different CPU architectures lead to cross-compilation and therefore necessitate an additional step for setting up a cross-compilation tool-chain (cross-compiler, cross-linker etc.). After the implementation on the target platform, the new service has to be tested in a test bed. Here, several limitations appear. The number of nodes in a test

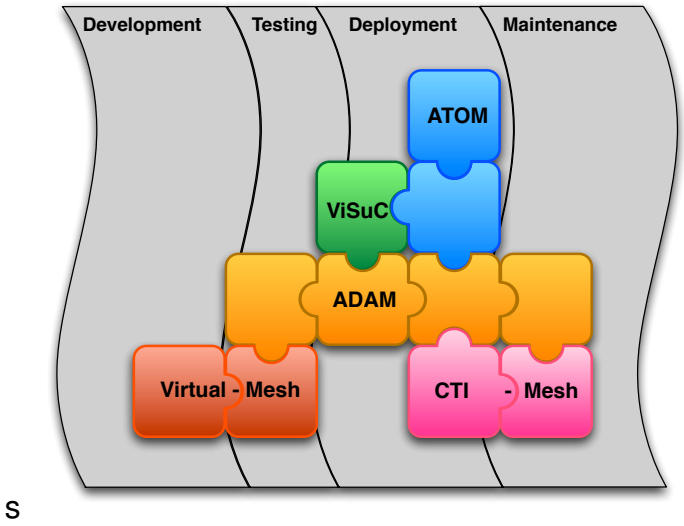
bed is usually limited and therefore does not support scalability tests. The management of the test bed is difficult. Configuration errors during tests may lead to unavailable nodes and therefore require on-site repairs of the nodes.
There are several problems the network engineer is facing during the deployment phase. Besides some software problems such as the stability of existing wireless network drivers, there are regulatory issues, finding accurate node sites and getting agreements with the land owners, and the weather that require weather sealing of the nodes, lightning protection, and storm-proof fastening of the masts. The challenge in the maintenance phase is having remote access to all nodes even in situations with faulty software updates or configuration error. After deployment the nodes are deployed in areas with restricted access (e.g., roof tops) or in hostile environment. Any software or configuration update can break the remote connectivity of the node in case of errors.

13.3 Contributions

Our five contributions to meet the challenges above are manifolded (see Figure 13.2). The first contribution is the architecture VirtualMesh, which fills the gap between development in a network simulator and testing in a test bed. It provides support for extensive predeployment tests. Our second contribution ADAM (Administration and Deployment of Ad-hoc Mesh networks) covers the creation of an operating systems tailored for the embedded devices in a WMN and the safe maintenance of the WMN. ViSuC (Visual Support for Constructions) - our third contribution - handles the deployment of a temporary WMN, which should be “as easy as winking”. The next contribution is CTI-Mesh, which is a feasibility study for WMNs as access network for a weather station. Finally, ATOM (Adaptive Transport over Multipaths) provides a solution for supporting real-time traffic in WMNs.

13.3.1 VirtualMesh: An Emulation Framework for Wireless Mesh Networks in OMNeT++

VirtualMesh [1, 2] faces the problems of simulations not covering operating system timings, time-consuming prototype testing in a real test bed, and the lack of large test beds for scalability tests. The key idea of VirtualMesh is to combine the features of simulation (scalability, fast and flexible testing) and testing on real systems (operating system timings, real software). It runs the real software on top of a simulated network. It provides a virtualization of a complete wireless mesh network by using host virtualization (XEN) for the mesh nodes and redirecting their wireless network traffic to a network simulator. For the Linux networking stack, the introduced virtual driver behaves as a normal wireless driver although it is performing the traffic redirection to the simulator. VirtualMesh is fully transparent



s

Figure 13.2: Own contributions in the WMN life cycle

to all the software above the network driver and provides flexible testing by the simulated network.

13.3.2 ADAM: Administration and Deployment of Ad-hoc Mesh networks

A deployment of a WMN requires an operating system, which is tailored for the embedded nodes. As the network can consists of different node types (e.g., Meraki Mini, ALIX, and WRAP), the operating system has to run on different platforms (CPU architecture). We need different cross-compilation tool-chains. But nevertheless, the same basic functionality for all node types should be provided. Moreover, the operating system should provide safe software update and configuration management facilities in order to avoid inaccessible nodes.
ADAM (Administration and Deployment of Adhoc Mesh networks) [3, 4, 5] provides solutions for a common operating system and safe software update and configuration management. ADAM includes a image build process which directly supports cross-compilation and individual build configuration for the three node types (Meraki Mini, ALIX, and WRAP). The ADAM build and management system is highly modular and can be easily extended with additional software packages or ported to a new node type. It fully supports IPv6, flexible management nodes, and ad-hoc routing protocols. ADAM further provides mechanisms for deployment and configuration a wireless mesh network. It guarantees the availability of the network despite configuration errors or faulty software updates by the mean of various fallback behaviors. It even provides a safe update of the operating system.

13.3.3 ViSuC: Video Support for Constructions

We have started the implementation of a temporary WMN based system to support video communication in large construction sites, which faces the problem of missing communication facilities at the time of electric installation. By providing video communication over an "easy-to-install" temporary WMN, the requirement of costly on-site visits by an electrical engineer is reduced.

13.3.4 CTI-Mesh: Wireless Mesh Networks for Interconnection of Remote Sites to Fixed Broadband Networks (Feasibility Study)

CTI-Mesh [6] provides actual deployment and maintenance experiences. We investigate whether wireless mesh networks (WMNs) are appropriate for connecting sensor networks or other devices deployed in remote areas, where no fixed network access is available, to a fixed broadband network. To support a variety of application scenarios, the WMN must meet reliability requirements and bandwidth in the 10 Mbps range over distances of several 10 km, e.g., by using directional radio transmission. During the project a WMN based on IEEE 802.11a/h (5 GHz) has to be deployed in the area of Neuchâtel and Payerne.
For the project we have mounted one mesh node on the roof of the University of Neuchâtel and another mesh node on the roof of SwissMeteo at Payerne. Intermediate nodes equipped with solar equipment (panels, chargers, and batteries) have been placed on the hills in the area to interconnect Payerne with the fiber network in Neuchâtel.
The network has been first tested on a small area in front of the weather station. Afterwards, the nodes have been moved to their final locations. First measurements show the necessity of a careful selection of the node locations, a good alignment of the antennas, and a strong tensioning of the antenna masts. In addition, aspects like birds using masts as raised blinds, storms, and the subsidence of tripod due to rain have to be considered during the deployment.
With the experiences gained from the deployment, we are now able to easily dimension further outdoor wireless mesh networks (approved equipment, possible distances, planning and setup time).

13.3.5 ATOM: Adaptive Transport Over Multipaths

Real-time communication in WMN is challenging. Due to the erroneous nature of the wireless medium, individual links are varying heavily in their quality of service. This affects real-time communications such as video conferencing by outages, artifacts, or stumbles. ATOM (Adaptive Transport over Multipaths) [7, 8]

reduces the problems of real-time transmissions over WMNs by using path diversity and multi-stream coding. This works as the characteristics of multiple paths are usually uncorrelated, i.e., the delay, jitter, and loss rate of the paths differ a lot from each other. Therefore, the transmission over multiple paths can be used to compensate for the dynamic and unpredictable nature of WMNs. In order to exploit this path diversity for improving the quality of the real-time transmission, a robust multi-path routing protocol and a mechanism for selecting appropriate coding and path allocation for the given network conditions are needed. At session establishment, ATOM decides on the used parameter set (encodings, paths etc.) considering current network conditions and collected historic data. After session establishment, the effect of this decision is continuously monitored and if necessary adapted.

13.4 Conclusion and Outlook

We provide a manifold framework for supporting a WMN during its life cycle. The key modules provide the following functionality: operating system builder, safe software and configuration updates (ADAM), fast prototyping and testing, scalability test of real implementation (VirtualMesh), “as easy as winking” deployment of a temporary WMN (ViSuC), deployment experiences, real deployment (CTI-Mesh), and support for real-time applications in WMNs (ATOM). During the next months, the finishing implementation and evaluation of the modules is envisioned.

References

[1] T. Staub, R. Gantenbein, and T. Braun, “Virtualmesh: An emulation framework for wireless mesh networks in omnet++,” in *The 2nd International Workshop on OMNeT++ (OMNeT++ 2009) held in conjuction with the 2nd International Conference on Simulation Tools and Techniques, Rome, Italy*, March 2009.

[2] T. Staub, R. Gantenbein, and T. Braun, “VirtualMesh: An emulation framework for wireless mesh networks in OMNeT++,” *Simulation: Transactions of the Society for Modeling and Simulation International*, submitted 2009.

[3] T. Staub, D. Balsiger, M. Lustenberger, and T. Braun, “Secure remote management and software distribution for wireless mesh networks,” in *7th International Workshop on Applications and Services in Wireless Networks (ASWN 2007), Santander, Spain*, pp. 47–54, May 2007.

[4] D. Balsiger and M. Lustenberger, “Secure remote management and software distribution for wireless mesh networks.” Computer science project, September 2007.

[5] D. Balsiger, “Administration and deployment of wireless mesh networks,” Master’s thesis, University of Bern, April 2009.

[6] T. Staub, M. Brogle, K. Baumann, and T. Braun, “Wireless mesh networks for interconnection of remote sites to fixed broadband networks,” in *Third ERCIM Workshop on eMobility, University of Twente, Enschede, The Netherlands*, pp. 97–98, May 2009.

[7] T. Staub and T. Braun, “Atom: Adaptive transport over multipaths in wireless mesh networks,” in *2nd ERCIM Workshop on eMobility, Tampere, Finland*, May 2008.

[8] T. Staub and T. Braun, “Supporting real-time communication in wireless mesh networks,” in *1st Workshop on Wireless Broadband Access for Communities and Rural Developing Regions - WIRELESS4D’08 held at 1st International Conference on M4D , Karlstad University, Sweden*, Karlstad University Press, December 2008.

14 EAP-TPM - A new authentication method for 802.11 based networks

Carolin Latze, University of Fribourg
carolin.latze@unifr.ch

14.1 Motivation

Public wireless networks require another type of user authentication than private networks at home. It is usually sufficient to have any type of encryption like WPA [1] or WPA2 without a special user authentication in private wireless networks. The situation is different for public hotspots since those are usually operated by an internet service provider (ISP) that needs to identify his users for legal and accounting reasons. Common authentication methods for that purpose are captive portals, EAP-TLS [2], and EAP-SIM [3], each with their specific advantages and disadvantages discussed in the remainder of this section.
Captive portals use a standard web-browser as authentication device by redirecting the first DNS request to the portal. An example of such a portal is shown in Figure 14.1. The page is obviously rather full with advertisements and other contents and will be hard to view on a browser of an embedded device like a mobile phone. Beside the bad scalability due to the packet filtering, captive portals are not really comfortable for the user. The reason why they are usually deployed in every hotspot is that they provide a good fall-back for unregistered users that can use their credit card to get temporary access.
The EAP-TLS [2] protocol provides mutual authentication based on X.509 certificates, which makes it a very secure protocol. However the need for X.509 certificates on the client side makes it hardly usable for normal users since the certificate requesting process is terribly complicated. Figure 14.2 shows an example certificate request form as it is used today. Even for an experienced user it is not trivial to fill in the required fields, a normal user will be completely lost. Furthermore in order to make EAP-TLS secure, the user's private key should be stored in a tamper resistant device making it even more complicated for the user.
In order to circumvent the need for X.509 client certificate but with a similar level of security, EAP-SIM [3] has been introduced and deployed. SIM cards are authentication devices users are used to. They allow for a comfortable authentication protocol but obviously EAP-SIM needs a SIM slot in the user's mobile device. Although that is given for mobile phones and some notebooks there are still a lot of device without SIM slot. There are already solutions where the user connects his mobile phone to the notebook to use the mobile phones SIM slot, but as SIMs only allow for single sessions, connecting his phone to his notebook for WLAN authentication prevents the user from receiving or making any phone calls. However

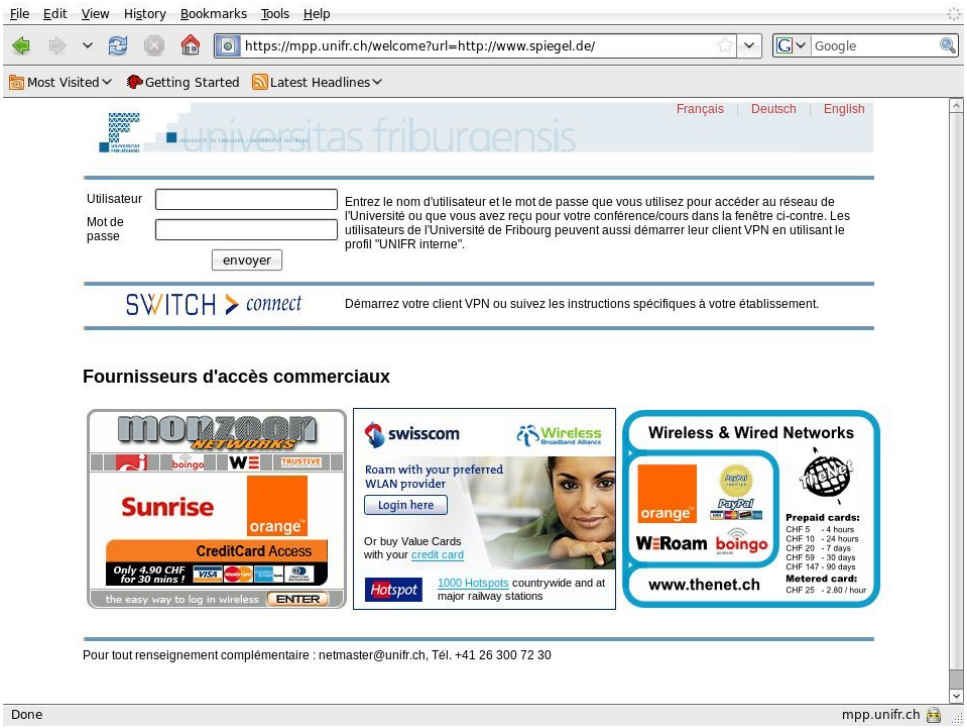


Figure 14.1: Captive Portal Page at the University of Fribourg

since EAP-SIM is a lot more comfortable from the user's point of view, hotspots in Switzerland implement EAP-SIM rather than EAP-TLS.

Since all three authentication methods mentioned above have problematic disadvantages, a new authentication method based on trusted computing has been introduced at the IETF in 2009 [4] that tries to overcome all the problems mentioned above.

14.2 Trusted Computing

The Trusted Computing Group (TCG) [5] is a standardization body formed to standardize the trusted platform module (TPM) and other trusted computing building blocks. The standard for the TPM that is used in the new authentication method has been written in 2002 and specifies a tamper resistant hardware token providing secure storage for hashes and keys. Furthermore it provides some cryptographic functions that are security critical. However, the TPM is no cryptographic co-processor. In fact the TPM is a rather slow module. The nice features of the TPM needed in the new authentication protocol are the possibility to identify it uniquely worldwide and the so called TPM identities. TPM identities are X.509 certificates bound to a certain TPM that can be used for authentication and attes-

Figure 14.2: Certificate Request Form of `http://www.swissdigicert.ch`

tation purposes.

Figure 14.3 shows the process of requesting a new TPM identity. The three left most entities *TPM*, *TSS*, and *Software* are located on the client, whereas the *PCA* is located somewhere in the network. The *trusted software stack (TSS)* is the library communicating directly with the TPM. It further implements functions that are not that security critical and can therefore be implemented in software only. The *software* is the user interface on the client PC. The authority that issues a new identity is a special certificate authority called *privacy CA (PCA)*.

The first step during the identity retrieval process is `TPM_MakeIdentity` which creates a so called *identity binding*. The *identity binding* is a signed collection of the public identity key and its name (chosen by the TPM owner) and the name of the *PCA* the user wants to request the identity from. The next (software) step is to collect all the certificates needed by the *PCA* to evaluate the request: the endorsement credential, the platform credential, and the conformance credentials. Those three credentials together with the *identity binding* are put together during `TPM_CollateIdentityRequest` and sent to the *PCA*. The *PCA* evaluates the data and issues the new identity. In order to protect that identity against tampering, the *PCA* encrypts it using a symmetric key that will be encrypted using the identity's public key. After having received the new identity it is up to the TPM to verify it. In order to do so, it decrypts the symmetric key and - if the decryption was success-

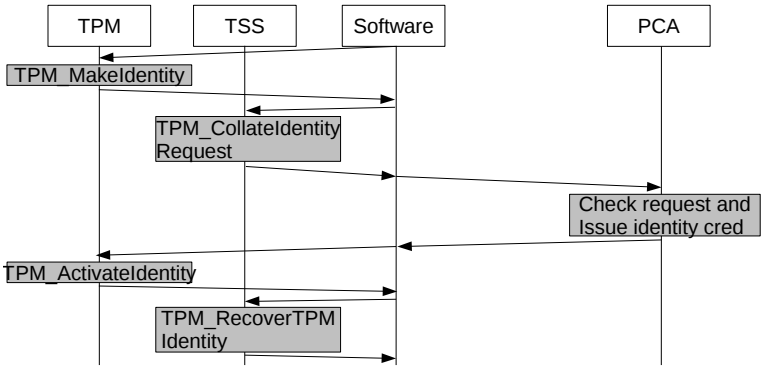


Figure 14.3: TPM Identity Retrieval

ful - releases the key to the host platform (`TPM_ActivateIdentity`). The last step is done at the *TSS*. `TPM_RecoverTPMIdentity` decrypts the identity certificate using the symmetric key. The *software* is now able to use the identity certificate for further applications.
As the TPM identifies itself during this process (it may also hold a lost of acceptable *PCAs*), there is no need for user interaction.

14.3 EAP-TPM

EAP-TPM as specified in [4] makes use of the fact that TPMs become more and more ubiquitous in consumer hardware. Together with the TPM identities they allow for a secure and user-friendly authentication scheme. EAP-TPM is based on EAP-TLS with only slightly modifications. The first modification is that the signing and hashing during the TLS handshake is done on the TPM. The second is the need for other certificate checks on the authentication server. It is necessary that the server verifies the certificate according to the specification released by the TCG [5]. EAP-TPM behaves slightly different for TLS 1.2 and TLS prior to 1.2. TLS 1.2 [6] uses standard signatures during the TLS handshake which allows to use the signing only identity certificate directly. Unfortunately, TLS prior to 1.2 [7] uses a non standard concatenation of an MD5 and a SHA1 hash during its handshake which requires some workaround in order to use the identity certificate

anyway. The EAP-TPM mode for TLS prior to 1.2 uses the identity certificate to create a so called certified key. That key is bound to the TPM and its identity and requires no additional user interaction to create it. Afterwards a self-signed certificate has to be created using that certified key which will then be used during the handshake. In order for the authentication server to be able to verify the self signed certificate a new TLS handshake message will be specified (following [8]) that allows to sent the identity certificate during the handshake too. The server has then all the information he needs to verify the client.
A proof of concept implementation of EAP-TPM can be downloaded at [9]. This implementation supports TLS prior to 1.2 only and makes use of a verification service holding all the identity certificates instead of using the supplemental data message.

14.4 Conclusion and Outlook

The EAP-TPM proof of concept implementation has shown promising results leading to a first real-world reference implementation done at Swisscom [10]. The standardization process for EAP-TPM is still in progress at the IETF [4].

References

[1] “IEEE Standard for Information technology - Telecommunication and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” 2007.

[2] D. Simon, B. Aboba, and R. Hurst, “The EAP TLS Authentication Protocol.” RFC 5216, 2008.

[3] H. Haverinen and J. Salowey, “Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM).” RFC 4186, 2006.

[4] C. Latze, U. Ultes-Nitsche, and F. Baumgartner, “Extensible Authentication Protocol Method for Trusted Computing Groups (TCG) Trusted Platform Modules.” Work in Progress, 2009.

[5] “http://www.trustedcomputinggroup.org,” 2009.

[6] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2.” RFC 5246, 2008.

[7] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.1.” RFC 5246, 2006.

[8] S. Santesson, “TLS Handshake Message for Supplemental Data.” RFC 4689, 2006.

[9] C. Latze, “EAP-TPM Proof of Concept Implemenation.” http://diuf.unifr.ch/people/latzec, year = 2008.

[10] C. Latze and J. Hiller, “EAP-TPM A New Authentication Protocol for IEEE 802.11 Based Network.” Demonstrations of the IEEE Conference on Local Computer Networks (LCN), 2009.

15 Modularization of the NIO Framework and Misuse prevention in PGA

Ronny Standtke, University of Fribourg
ronny.standtke@unifr.ch

15.1 Modularization of the NIO Framework

15.1.1 Introduction to NIO

A new input/output (NIO) library was introduced with Java 1.4. It provides high-speed, block-oriented I/O and takes advantage of low-level optimizations in a way that the original I/O package could not, without using native code. But actually creating network applications with the NIO library has always been very difficult because the paradigm shifted from synchronous to asynchronous I/O. A programmer using NIO suddenly has to deal with a lot of complex new issues, e.g. incomplete read/write operations, data buffering, buffer overflows, buffer flipping, managing channel interest operations, queue handling, thread pools, thread synchronization, etc.

A programmer first has to take care of all this issues and must implement a lot of boilerplate code just to get a little working NIO network application. The issues are even doubled when using NIO with SSL. Things that have been very simple with the classic I/O (e.g. waiting for an SSL handshake to complete) are now extremely complicated.

15.1.2 Introduction to the NIO Framework

The NIO Framework is a library on top of NIO that hides most of the complexity of plain NIO. With the NIO Framework one can implement high-performance Java network applications without having to deal with all the nasty details of NIO. The issues above are resolved while the performance is preserved. In addition to that the NIO Framework provides many I/O building blocks for application programmers, e.g. dummy traffic, traffic shaping and traffic accounting.

The NIO Framework is an Open Source project and can be downloaded from `http://nioframework.sourceforge.net`.

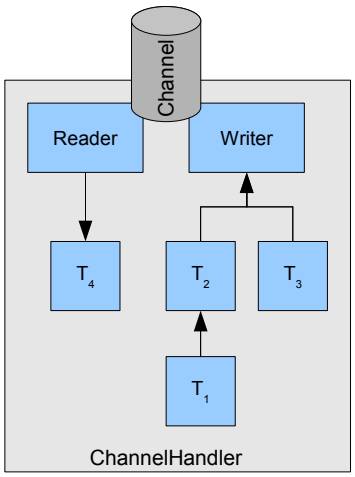


Figure 15.1: I/O Transformation example

15.1.3 Modularization

When application data units (objects, messages, etc.) have to be transmitted over a TCP network connection, they have to be transformed into a serialized representation of bytes.
There are many ways to represent application data and there are also many ways to serialize data into a byte stream. Therefore, there are countless transformations between application space and network space imaginable.
The first approach to this problem in the NIO Framework was to provide an extensible hierarchy of classes, where every class dealt with a certain transformation (e.g. string serialization, SSL encryption). This architecture turned out to be very simple and efficient. The downside of this approach was that every combination of transformations required its own implementing class. Changing the order or composition of transformations was very difficult and much too inflexible for a generic framework.
The second and current approach to message transformation is to implement a set of transformer classes were each class offers just a certain transformation. An application programmer can put these transformer classes together into a hierarchy of almost arbitrary order. Almost no programming effort is required besides assembling the needed classes of the transformation hierarchy in the desired order.
A diagrammatic example of the I/O transformation architecture is shown in Figure 15.1.
The shapes T_x are the transformation classes. When writing to a channel, the ChannelHandler hands the application level data units to one of the input transformation classes T_1, T_2 or T_3 (depending on the type of input it just accepted). Every transformation class transforms the data and hands it over to its next transformer until it reaches the Writer, which writes the final byte stream to the channel

and handles many channel specific problems, e.g. incomplete write operations. When reading from a channel, the Reader handles the channel specific problems, e.g. connection closing and read buffer reallocations. After reading a byte stream from the Channel, the Reader passes the data to T_4, which transforms the data. The ChannelHandler can get the application level messages from T_4.
There are four basic I/O models for the transformation classes T_x. In ascending order of complexity they are:

- **1:1** (one type of input, one type of output)
- **1:N** (one type of input, different types of output)
- **N:1** (different types of input, one kind of output)
- **N:M** (different types of input, different types of output)

Every model is valid insofar as one can establish a fully functional transformation hierarchy with any of these I/O models. While the 1:1 model would be the most simple one, transformation classes of the N:M model would have the highest flexibility. The interesting thing to note here is that with respect to flexibility every transformation class of the more complex models can be replaced by chaining several transformation classes of the 1:1 model. While trying to implement prototypes for all models above it became clear that the most simple API was provided by using Java Generics and the 1:1 model. Another advantage of the 1:1 model is the encouragement of code reuse, because every transformation should be implemented in a separate class.
The elegance and simplicity comes at the small price of an almost immeasurable performance loss. Currently, Java Generics are implemented by type erasure: generic type information is present only at compile time, after which it is erased by the compiler. The compiler automatically inserts cast operations into the byte code at necessary places which may cause a tiny performance loss. Using the 1:1 model results in slightly longer transformation chains, more involved objects and more locking and unlocking when passing data through a transformation hierarchy.

Forwarders and Transformers

For the current version of the Java NIO Framework the following set of forwarders and transformers (forwarders with a certain transformation) have been created. There are basic forwarders and composite forwarders. Basic forwarders do not depend on other forwarders. Basic forwarders have been created e.g. for buffering, prefixing, framing, encrypting and decrypting data and forwarders that deal with all the NIO details when reading from or writing to a Channel. The implemented forwarders for generating and parsing data streams that are padded with dummy traffic are composite forwarders that use the available basic forwarders and transformers as building blocks.

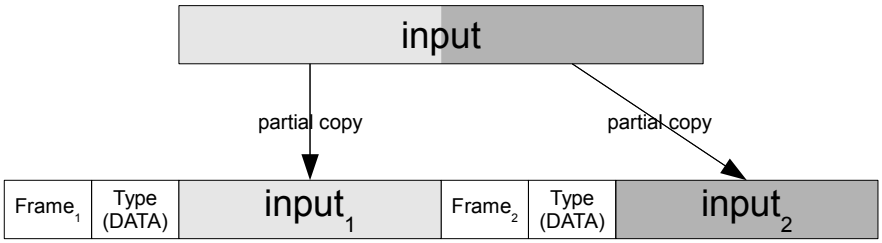


Figure 15.2: buffer splitting with partial data copies

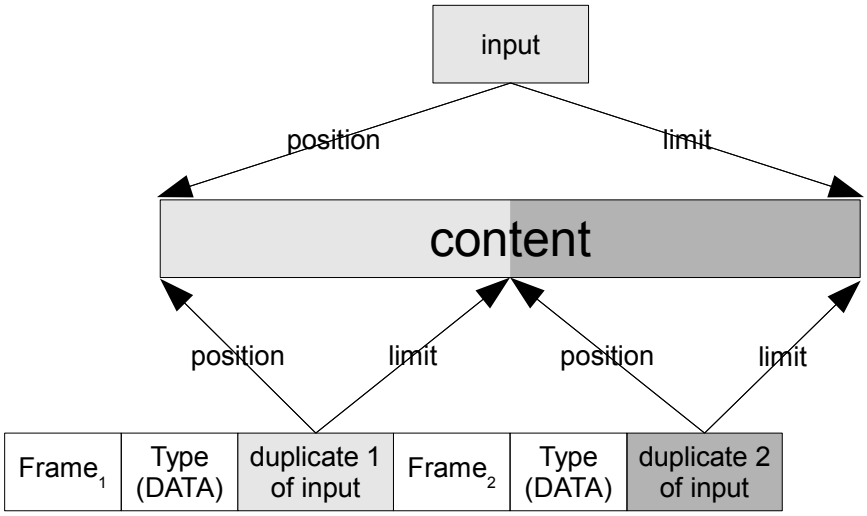


Figure 15.3: buffer splitting with buffer duplication and content sharing

Prevention of redundant copy operations

When using a complex hierarchy of forwarders and transformers data is buffered at different places in that hierarchy. When forwarding data through this hierarchy it happens very often that only a part of the buffered data has to be forwarded. The classical and simple approach to this problem is to create safety copies of the individual parts (see Figure 15.2). Unfortunately, this approach lowers the I/O performance of the whole hierarchy and uses unnecessary amounts of RAM.

For storing data in its forwarders and transformers, the Java NIO Framework uses Java ByteBuffers. These ByteBuffers have an interesting feature: They can be duplicated without copying data around in memory. Instead, the content of duplicated buffers will be shared but the context information (position, limit, capacity, ...) can be set independently. By creating ByteBuffer duplicates and setting their positions and limits in subsequent order, data packages can be broken up into smaller parts and forwarded through a hierarchy of forwarders and transformers without creating unnecessary partial copies (see Fig 15.3).

15.2 Misuse prevention in PGA

15.2.1 Introduction to PGA

There are several anonymity architectures for Internet communication in use today. They are either unsafe or very complex. The PGA (”Pretty Good Anonymity”) architecture is an anonymity service that aims to provide a high level of protection and is still simple enough to enable high-bandwidth, low-latency Internet communications. The PGA architecture uses a single-node anonymity service provider in combination with dummy traffic.

15.2.2 Misuse prevention

An anonymity service provider operates in an area of conflict. On one side the provider wants to deliver a trustworthy service but on the other side misuse of the service should be prevented. To be able to prevent misuse, the anonymity service provider must execute a very detailed analysis of the traffic before serving it, which contradicts the first goal of providing a trustworthy service.
A reasonable middle ground is ”misuse discouragement”, i.e. not to prevent misuse but to be able to detect it and react accordingly. For the PGA project this middle ground is reached by optionally logging communication circumstances. When using PGA, the anonymity service provider can switch on a logging function that records:

- time-stamp
- source IP
- destination host name
- resolved destination IP
- destination port
- success or failure of the connection attempt

There are several security mechanisms in the PGA implementation to make this function transparent for the users of the anonymity service and also to protect the anonymity service provider against itself:

User notification: Whenever the anonymity service provider switches the logging feature on or off it is signalled to all users of the service and displayed in the GUI of the PGA client program.

Encryption: The recorded logging data is very sensitive information. Therefore it must always be stored on an encrypting file system. This protects the

data in case of physical theft of the storage media. While the records are written to the file system, the encrypting file system is in an ”open” state, i.e. the system processes can read from and write data to it. To protect the logging data against compromization by hacking into the PGA server itself, it can be encrypted with a public key. It is recommended to use an external public key for this task, e.g. from a notary.

Filtering: It should not be necessary to record all connection attempts that are made over the anonymity service. Therefore the PGA implementation allows to configure regular expressions that define what will be logged and what not.

Retention period: The recorded logs are automatically removed after a configured retention period.

The definition of filters and a retention period are directed to the well established principle of data avoidance and data economy.