ANA Project
Autonomic Network Architecture

Sixth Framework Programme
Priority FP6-2004-IST-4
Situated and Autonomic Communications (SAC)

Project Number: FP6-IST-27489

Deliverable D.3.4v1

Self-optimisation in ANA (v1)

Authors
(alphabetical order)

Simon Balon, Sandrine Calomme, Guy Leduc, Jin Xiao, Maxwell Young

Version: 1.0

ANA Project
Autonomic Network Architecture

Project Number	FP6-IST-27489
Project Name	ANA - Autonomic Network Architecture
Document Number	FP6-IST-27489/WP3/D.3.4v1
Document Title	Self-optimisation in ANA (v1)
Workpackage	WP3
Editor	Guy Leduc (ULg)
Authors	Simon Balon, Sandrine Calomme, Guy Leduc (ULg) Jin Xiao, Maxwell Young (UWaterloo)
Reviewer	Ioannis Stavrakakis (NKUA)
Dissemination level	Public
Contractual delivery date	31^{st} Dec 2007
Delivery date	13^{th} Feb. 2008
Version	Version 1.0

Abstract:

We describe optimization issues encountered in large distributed systems, when every compartment can run its own optimization algorithm. Firstly we consider compartments forming a structured but flat topology, and we show how a compartment may reach a better optimum if it is aware of the intercompartment level. We also address possible instabilities resulting from the impact a compartment optimization may have on others. Secondly we consider compartments laid over one another, and we show how an overlay compartment can optimize its topology with no cooperation from the underlay compartment. We also address this kind of multilayer optimization using a cooperative approach to optimize the placement of data at the overlay level.

Keywords:

Compartment optimization, intercompartment awareness, instabilities, game theory, overlay optimization, cooperation

Executive Summary

In the ANA environment, self-optimization refers first to the ability of a compartment to self-optimize. In this document we have addressed three kinds of representative optimization problems, namely traffic balancing/engineering, topology control, and data placement. However when many compartments co-exit, which is the case in the ANA architecture, simply running independent optimizations in distinct compartments may not necessarily give the best results and may even lead to instabilities. And this is true for peer compartments in a flat structure, as well as for compartments laid over one another. These two structures are therefore considered in this document. Yet another issue addressed in this deliverable is the level of co-operation between compartments, or the level of intercompartment-awareness when running compartment-local optimizations.

Based on these concerns this document has four main parts describing four kinds of optimization problems encountered in large multicompartment distributed systems. In the first two parts we consider a federation of autonomous compartments forming a large but flat intercompartment system and we show how a compartement may reach a better optimum if it is aware of the intercompartment level. This is illustrated with a traffic engineering example. In the same context of peer compartments, we also address possible instabilities resulting from the impact the optimization of a compartment may have on other compartments, and we propose to use game theory to model and reason on this problem. In the third and fourth parts we consider compartments laid over one another, and we show how an overlay compartment can optimize its topology so as to stay close enough to the underlay topology, with little or no cooperation between both levels. On the other hand we also address this kind of multilayer optimization problem using a cooperative approach to aid compartments in reducing intercompartmental traffic by having compartments optimize the placement of data in the network so as to bias traffic.

This deliverable synthesizes the results achieved so far in task 3.2 (Self-optimization) of the ANA project, and is voluntarily kept short. Chapter 2 has been accepted for publication [1]. A preliminary work of chapter 4 has been published [2] and an extended version of it has been accepted for publication in [3]. Finally, two papers based on chapters 3 and 5 are in preparation.

Table of contents

Chapter 1

Introduction

In accordance with the ANA architecture, we see ANA environments as a (possibly large) set of fully autonomous compartments interconnected to each other to form an intercompartment topology. Some compartments may be connected to other ones through some gateway nodes. Some compartments may also be laid over one or several other compartments.

Optimizing such a distributed system raises a number of issues. In accordance with a compartment's full autonomy, we consider that every compartment may have its own (self-)optimization mechanisms (e.g. a traffic engineering algorithm that optimizes the traffic crossing this compartment according to some local cost function). This optimization may be purely intra-compartment, which means that the compartment is unaware of any possible impact its own optimization may have on other compartments and therefore does not take this impact into account, even if there is one, thus leading to a possible underoptimization of its own compartment and/or possible instabilities. In the best case, the intra-compartment optimization may be inter-compartment aware. This means that the compartment can anticipate the consequences of intra-compartment optimizations and find a better optimum and/or improve stability.

This first important issue is addressed in chapters 2 and 3. In chapter 2 we study the problem of intra-compartment traffic optimization as a typical optimization problem. We show that if the compartment only knows its intra-compartment traffic matrix as input, a typical traffic optimizer, such as a link cost optimizer, will not necessarily anticipate that changes in internal link costs may also change the exit point of some traffic and therefore change the intra-compartment traffic matrix, which was the input of the optimizer! A first consequence is that the resulting operating point is not the expected one, and the compartment is underoptimized. As a solution to this problem we propose a method to find a better optimum by raising the awareness of a compartment to some intercompartment routing rules.

However, the change of the exit points for some traffic also changes the input traffic of the downstream compartments. This cascading effect may even reach again a previous compartment, thus forming loops leading to possible instabilities in the combined optimization of the whole system. In chapter 3 we address this kind of distributed

optimization problems among fully autonomous compartments in the setting of game theory. We investigate the concept of self-stability by establishing realistic analytical models to determine the stability of ANA-like environments and methods of convergence. The goal is to propose practical implementation techniques and compartment designs that facilitate self-stability.

So far we have considered that the compartements were sitting side by side to form a structured but flat topology. Other issues appear when compartments can be laid over one another, thus forming multilayered architectures. To illustrate this we can consider an infrastructureless wireless compartment running its own topology control mechanism, which tries to optimize its topology in order to provide better capacity. Indeed if the wireless ranges of the nodes are too short the compartment may not be a connected graph, but if the ranges are too wide then nodes create too much interference with nearby transmissions, thereby decreasing the capacity of the compartment. A topology control algorithms tries to adapt the ranges dynamically in a distributed way to ensure connectivity and optimize the capacity of the compartment. Now consider another compartment laid over this wireless compartment. The overlay compartment typically consists of a subset of the underlay nodes forming a community running a particular protocol. Suppose the overlay compartment also runs a topology optimization algorithm. This means that overlay nodes will not form a full-mesh topology, but will rather select a few nodes as neighbours and reach other nodes through one or several intermediate overlay nodes. To optimize the overlay compartment, it is much better to have some awareness of the underlying topology. For example, if the overlay knows the number of wireless hops to reach other overlay nodes, it will preferably choose close neighbours. In other words, the overlay should try and map the overlay topology to the underlay topology. However, when the underlay topology changes over time, due to an underlying topology control algorithm or due to the mobility of nodes, then the overlay should track those changes to remain optimal. In chapter 4 we study this problem and propose an overlay topology control algorithm that finds a good trade-off between properties like path stretch, delay and bandwidth consumption in the more challenging environment of mobile ad hoc underlay compartments.

In the previous example no cooperation is assumed between the underlay and the overlay. Another kind of multilayer optimization where cooperation is considered is the data placement problem in peer-to-peer (p2p) overlays. The rise of p2p applications poses a new challenge for underlay compartments whose respective infrastructures are now facing increased and sustained traffic loads. These problems are not necessarily due to the number or size of files being shared between peers, but rather result from topological discrepancies between the overlay and underlay networks. Clearly, p2p traffic across compartments should be minimized so as to utilize available bandwidth effectively. However, there is no easy solution to this problem in the flexible and completely decentralized ANA architecture. In chapter 5, we propose a cooperative approach to aid compartments in reducing inter-compartmental traffic by having compartments optimize the placement of data in the network so as to bias traffic.

1.1 Scope of Deliverable

In the ANA environment, self-optimization refers first to the ability of a compartment to self-optimize. In this document we have addressed three kinds of representative optimization problems, namely traffic balancing/engineering, topology control, and data placement. When many compartments co-exit, which is the case in the ANA architecture, simply running independent optimizations in distinct compartments may not necessarily give the best results and may even lead to instabilities. And this is true for peer compartments in a flat structure, as well as for compartments laid over one another. These two structures are therefore considered in this document. Yet another issue addressed in this deliverable is the level of co-operation between compartments, or the level of intercompartment-awareness a compartment has when it runs a compartment-local optimization.

1.2 Structure of Document

Chapter 2 proposes an intercompartment-aware compartment optimization that improves classical purely intracompartment algorithms. It will appear in [1]. Chapter 3 studies instabilities resulting from independent optimizations in different autonomous compartments, and investigates the concept of self-stability in the setting of game theory. Chapter 4 proposes a self-optimized topology control algorithm for compartments laid over mobile underlay compartments. An extended version has been accepted for publication [3]. Chapter 5 proposes a method to optimize the placement of data in the overlay compartment so as to reduce traffic between underlay compartments.

Chapter 2

Intercompartment-aware compartment optimization

Intracompartment traffic engineering consists in routing traffic in an optimal way from ingress nodes to egress nodes in a compratment. If shortest path routing is used, the only way to optimize the traffic paths is by finding an appropriate set of link weights that minimizes a given objective function. For example, if this objective is to minimize the total (equivalently the average) link *load*, a solution consists in assigning unitary weights to all links in the network. On the other hand, to minimize the average link *utilization*, the solution consists in choosing link weights that are inversely proportional to the link capacities. Note that these two examples are representative of traffic-independent link weights settings, i.e. they minimize their respective objectives for *every* possible traffic matrix.

For other objective functions (e.g. minimizing the maximum link load or utilization), the optimal choice of link weights usually depends on the traffic matrix. Therefore in its simplest form the resolution of this optimization problem needs to take as inputs (1) the topology with unknown link weights, (2) the chosen objective function, and (3) a traffic matrix, which specifies the amount of traffic between every pair of ingress/egress nodes [4]. This optimization problem is NP-hard and good local-search heuristics are thus needed to find a set of link weights that reasonably minimizes the objective function in a reasonable time.

However this approach is unaware of the interdependence between intracompartment and intercompartment routing protocols. If an "early exit" rule is present in the intercompartment routing protocol (as it is the case for BGP), the real traffic demand is actually an intercompartment traffic matrix (interTM), while the intracompartment traffic matrix (from ingress to egress nodes - intraTM) is only the result of applying intercompartment routing decisions on the interTM. Even if we consider that the interTM and the intercompartment routes are invariant, the intraTM may still vary if some link weights are changed inside the compartment. This is typically be due to the early exit decision rule. Given the rationale of such a rule, it is indeed likely that ANA intercompartment routing protocols will implement something similar.

2.1 Related Work

A first Link Weight Optimizer (LWO) for a given intraTM has been proposed by Fortz et al. in [4]. It is based on a tabu-search metaheuristic and finds a nearly-optimal set of link weights that minimizes a particular objective function, namely the sum over all links of a convex function of the link loads and/or utilizations. This problem has later been generalized to take several traffic matrices [5] and some link failures [6] into account. In our LWO we reuse the heuristic detailed in [4], but we have adapted this algorithm to consider the effect of early exit routing. All the later improvements to this algorithm (i.e. multiple traffic matrices, link failures) could be integrated in our new LWO in a similar way.

The fact that the intracompartment TM is not the correct input for many Traffic Engineering problems had already been pointed out in [7, 8] by Feldmann et al., who suggested to consider the set of possible egress links in the traffic matrix. In [9] several extensions to the classical LWO problem are briefly described by Rexford, including a sketch of a method that resembles ours. Our work is in line with this recommendation, as we connect several equivalent egress nodes to a single virtual node representing the destination, but our work proposes a complete method to solve the link weights optimization problem, applicable to intracompartment and peering links, and we demonstrate its efficiency on an operational network. In [7] some methods to compute traffic matrices from netflow traces are also presented, which are reused in this paper.

Cerav-Erbas et al. have already shown in [10] that the link weights found by a LWO may change the intracompartment TM considered as input. In that paper they also show that applying LWO recursively on the resulting intracompartment TM may not converge. They propose a method that keeps track of the series of resulting TMs and at each iteration they optimize the weights for *all* the previous resulting intracompartment TMs simultaneously. However, they do not consider the general problem with multiple exit points for each destination, let alone taking advantage of it.

2.2 An Intercompartment-aware link weights optimizer

A network is modeled as a directed graph, $G = (N, L)$ whose vertices and edges represent nodes and links. The basic intracompartment topology is composed of all the nodes and links that belong to the compartment. We consider two disjoint categories of destination addresses. The *single-egress addresses* are those destinations for which only one next hop is preferred by the intercompartment routing protocol, regardless of any early exit criteria. The *early exit destinations* are all the other addresses for which the intercompartment routing protocol will use the early exit criterion to select the egress node in the compartment. The traffic forwarded to the *single-egress destinations* constitutes a (early exit invariant) intraTM, called TM_{invar}.

For every early exit destination we conceptually add a *virtual node* representing it. Then for every peering link on which early exit destinations have been announced, we

extend the intracompartment topology with a link+node pair representing this peering link and the neighboring router on the other side of this link. Finally we attach all these neighboring routers to the virtual node (representing the early exit destination) by adding *virtual links*.

Therefore we have three disjoint sets of edges in the topology: L_{intra} is the set of intracompartment links, L_{inter} is the set of intercompartment links, and $L_{virtual}$ is the set of virtual links. Similarly we split the nodes in the topology into three disjoint sets: N_{intra} is the set of routers from the local compartment, N_{neigh} is the set of border routers in neighboring compartments, and $N_{virtual}$ is the set of virtual nodes. Figure 2.1 shows such a topology. P_1, P_2, P_3 and P_4 are early exit destinations.

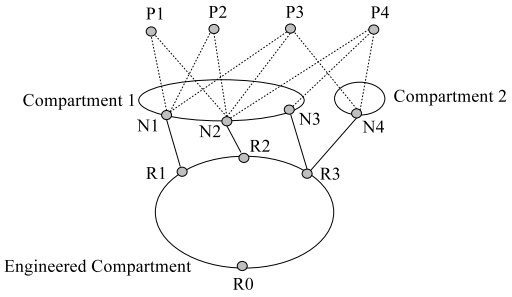


Figure 2.1: Topology with virtual nodes

Each virtual link ($l \in L_{virtual}$) has infinite capacity $c_l = \infty$ and a fixed weight $w_l = 0$. Every other link ($l \in L_{intra} \cup L_{inter}$) has a capacity c_l and a weight w_l. Let us note that intercompartment and virtual links are directed (toward the destination) as no transit via a virtual node is allowed.

The traffic will follow the shortest path(s) based on the link weights, which respect both the intracompartment routing protocol based on shortest paths and the intercompartment routing protocol's early exit rule. Once the paths are chosen, we can associate with each link l a load l_l, which is the proportion of traffic that traverses link l summed over all pairs of source/destination nodes. The utilization of a link l is $u_l = l_l/c_l$.

The goal of the LWO is then to find the set of link weights that minimizes our network-wide objective function based on the loads and/or utilizations of intracompartment and intercompartment links.

2.2.1 Aggregating destinations

The problem as formulated in the preceding section is not solvable in practice. Indeed the number of destination addresses in the routing table of a router would be too large. However all destinations that are reachable through exactly the same set of possible nodes $\in N_{neigh}$ can be aggregated (e.g., nodes P_1 and P_2 in figure 2.1 can be merged) as they are indistinguishable from an intracompartment routing perspective. This will drastically reduce the number of virtual nodes. Note that if n is the number of peering links of the AS, there can still be 2^n virtual nodes in the worst case. In practice however

it is much lower, as explained in [11]. Indeed routes are often announced with the same parameters on peering links with the same neighbor AS. For the operational network we have used as a case study, the number of peering links traversed by early exit traffic is 18. Out of 2^{18} possible different combinations of peering links, only 26 are actually observed!

We can still go one step further by taking the traffic destined for each aggregated virtual node into account. For example, in our case study we have noticed that no traffic is sent to 8 of them, and only a very small volume of traffic is sent to 13 others, thus leading to 5 nodes receiving 99.94% of the early exit traffic. So we can basically extend the intracompartment topology with these 5 virtual nodes without really losing accuracy. This is really significant for the practical efficiency of the LWO. More precisely, using 5 nodes instead of 18 reduced the average computation time of the algorithm from 582 to 140 seconds[1] without decreasing the quality of the provided solutions. Stated otherwise, the same computational budget would allow us to find a better solution (using more iterations) on the smaller topology.

2.2.2 Engineering intra- and intercompartment links

LWOs usually try and find a nearly optimal set of intracompartment link weights. An optimal set of weights is defined as a set of weights that associates the minimal value to a predefined objective function. The objective function is generally the sum over all the links of a convex function of the link load and/or utilization. In [4] they use a piecewise linear convex function of the link utilization and capacity (ϕ_l for link l): $\phi = \sum_{l \in L_{intra}} \phi_l$, where L_{intra} is the set of intracompartment links. We reuse this network-wide objective function, while others could be used in the optimizer.

As intercompartment links are now part of the topology, we can include these links in the objective function. We are flexible with respect to the inclusion of these intercompartment links in the objective function by adding a parameter α which determines the relative importance of intercompartment links with respect to intracompartment ones. The new function is $\phi = \sum_{l \in L_{intra}} \phi_l + \alpha \sum_{l \in L_{inter}} \phi_l$.

The inclusion of intercompartment links in the objective function is a key advantage of our method as it allows the LWO to engineer these intercompartment links in addition to intracompartment ones. With a classical LWO there is no point in including intercompartment links in the topology and engineer them, because the intracompartment TM used as input pins down the egress node anyway, thus assigning the same load on the intercompartment links irrespective of the link weights. As we relax the constraints on the egress nodes, it seems natural to take advantage of it to also engineer the traffic on intercompartment links. With our method it suffices to include these links in the objective function.

[1] This is the average computation time over 14 runs on different TMs with 50 iterations per run. We have used 50 iterations because we have noticed that increasing this number did not significantly improve the quality of the solution found on this data. These simulation times are measured on an IBM computer eServer 325 with 2 AMD opteron 2GHz 64 bits processors and 2GB of memory.

2.3 Simulations on an operational network

To test our algorithm we have used the current Internet. We have mapped an ANA compartment on the AS (Autonomous System) concept, and we have used BGP as intercompartment protocol, which means that destinations are grouped into prefixes. We have tested our algorithm on real data of a multi-gigabit operational network that spreads over the European continent and is composed of about 25 nodes and 40 bidirectional intracompartment links. Link capacities range from 155Mbps to 10Gbps. It is a transit network that has two providers connected with about 10 intercompartment links, has other peer ASes connected with about 15 shared-cost links, and has more than 25 customer ASes, which are mainly single-homed. The total traffic exchanged is about 10 Gbps on average.

We had access to about one month of traces, one BGP dump per day and one sampled netflow file for each ingress router. With these data we have generated 2,512 aggregated interTMs (each matrix is an average over 15 minutes). This whole set of traffic matrices is representative of the traffic on the studied network. Some of these induce a low load on the network while some induce a high load[2].

The average number of prefixes is 160,973 of which 97.2% (156,407) are early exit prefixes. If we now take traffic into account, we have measured that these 97.2% amount to 35.6% of the traffic on average. This is still enough to have a significant impact on the link loads of the network. Over all recorded TMs, the peak value is 51.7% of the traffic and the minimal value is 24.6%. Another interesting fact is that on average 99.94% of early exit traffic is destined for the 5 biggest clusters of prefixes. The sets of intercompartment links giving access to each of these 5 clusters of prefixes are either all peering links to a neighboring AS (for 3 clusters), or a mix of peering links from two such ASes (for 2 clusters).

2.3.1 Intracompartment TE

We first compare a classical LWO (denoted *IntraLWO*) with our intercompartment-aware optimizer (denoted *EE-awareLWO* for Early-Exit-aware LWO). To execute *IntraLWO* we had to generate for each intercompartment TM the corresponding intracompartment TM where the early exit traffic is routed considering the present (i.e., non engineered) link weights. So these intracompartment TMs are those that would be measured in the network. For the comparison we have run both optimizers on all the 2,512 aggregated intercompartment TM. Optimizers consider weights in a range from 1 to 150. Figure 2.2 shows the maximal intracompartment link utilization (U_{max}) for some worst-case TMs.

We have run *IntraLWO* on every intracompartment TM, and computed the resulting maximal link utilization, assuming that the intracompartment TM remains invariant (thus ignoring early exit effects). In the sequel these values are denoted *IntraLWO-*

[2]The set of intraTMs built from the same BGP data and neflow traces is described in [12].

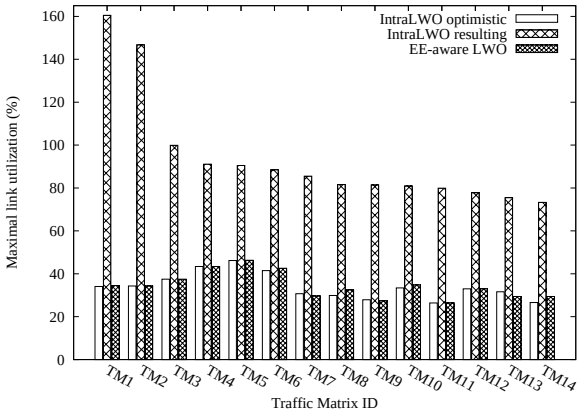


Figure 2.2: U_{max} values for some worst case TMs

optimistic. For this link weights setting, if early exit effects are taken into account, we get the resulting maximal intracompartment link utilization denoted *IntraLWO-resulting*. These are the real values that would be observed if the optimized link weights were installed in the network. These values are very different, and sometimes the resulting maximal utilization is even worse than the routing without link weight optimization (not present in the figure). Finally we have run our *EE-awareLWO* and we can see that the maximal link utilizations are very good. Figure 2.2 shows a selection of TMs providing the worst-case values for *IntraLWO-resulting*[3]. The average reduction of U_{max} from *IntraLWO-resulting* to *EE-awareLWO* over all TMs is 4.5%, but let us outline that the worst-case TMs do matter much more, because the main goal of our LWO is to filter out the unexpectedly bad link weights settings proposed by a classical LWO. In all cases the real minimal value of U_{max} achievable *in practice* are the values of *EE-awareLWO*, since the *IntraLWO-optimistic* are disqualified in the comparison.

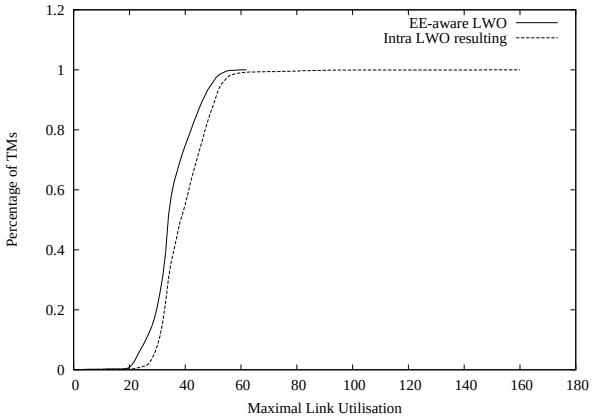


Figure 2.3: CDFs of U_{max} over all TMs for *EE-awareLWO* and *IntraLWO-resulting*

Figure 2.3 shows the CDFs (cumulative distribution functions) of the maximal link utilization over the 2,512 TMs for *EE-awareLWO* and *IntraLWO-resulting*. *IntraLWO-optimistic* is not depicted on the figure because it would be almost mixed up with *EE-*

[3]In this case we define worst case values as values of traffic matrices providing the highest intracompartment maximal link utilizations.

awareLWO. We can clearly see that *EE-awareLWO* is better than *IntraLWO-resulting*. Figure 2.4 gives the proportions of TMs per range of maximal link utilizations. In this figure we can see that *EE-awareLWO* takes advantage of the freedom of choice of the egress point(s) for early exit traffic. Indeed *EE-awareLWO* is slightly better than *IntraLWO-optimistic*. For example there are 3.4% less TMs in the [30, 40) range. This indicates that our optimizer can change the egress point of some early exit traffic to better engineer the network.

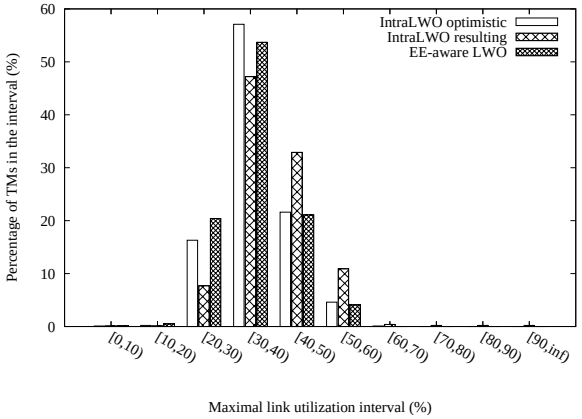


Figure 2.4: Proportions of TMs in each U_{max} interval

2.3.2 Intercompartment Optimization

One of the most innovative feature of our LWO is its ability to engineer traffic on the intercompartment links. We first analyse the maximal link utilizations of the intercompartment links with the present link weights. The average value of Intercompartment U_{max} over all TMs is 36.8%. This value can peak at 73.7%. We have selected the worst TMs in this respect[4] and run *E*E-aware LWO on them with intercompartment links in the objective function. The results are shown in figure 2.5 for the peak TM. The maximal intercompartment link utilization is reduced from 73.7% to 36.8% when using *E*E-aware LWO. It shows that the LWO can take advantage of early exit routing to also engineer traffic on intercompartment links.

We now show that the optimization of intercompartment links is not done at the expense of intracompartment links. To this end we have run *EE-aware LWO* with and without intercompartment links in the objective function ($\alpha = 1$ or $\alpha = 0$, see section 2.2.2) on the 50 TMs leading currently to the maximal intercompartment link utilization. Figure 2.6 presents the average intracompartment and intercompartment U_{max} values for these matrices. It shows that *EE-aware LWO* with all links in its objective function can optimize intercompartment links almost without impacting intracompartment links. The average intracompartment U_{max} value is indeed almost equivalent in both cases.

[4]Here worst case TMs means TMs providing the highest intercompartment link utilization with present link metrics.

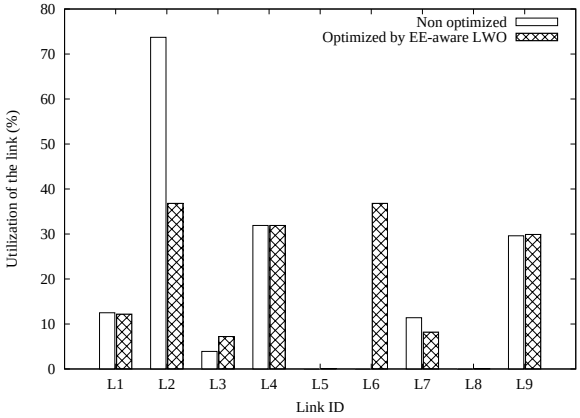


Figure 2.5: Intercompartment link utilizations

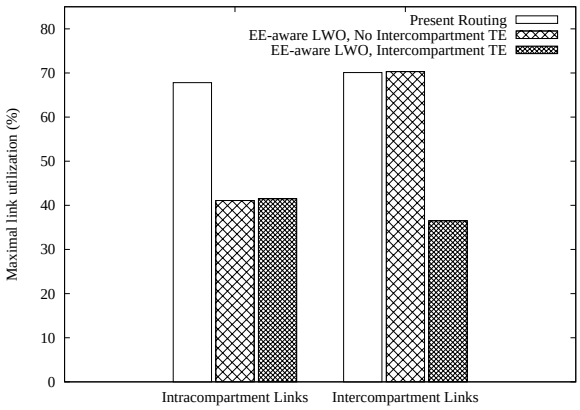


Figure 2.6: Combined Intra- and Intercompartment Traffic Engineering

2.4 Conclusion

We proposed a intercompartment-aware Link Weight Optimizer (LWO) that extends the classical (intracompartment) LWO to take into account early exit routing principle. The optimized link weights, if deployed, will actually give rise to the link loads expected by the optimizer, contrary to a classical (intracompartment) LWO that may lead to unexpectedly high loads on some links when changing weights impact the intraTM. In practice the method only requires to extend the intracompartment topology with a limited number of virtual nodes and links, which preserves scalability, as shown on an operational network used as a case study. The aggregated interTM associated with this extended topology replaces advantageously the classical intraTM as input to the LWO. On this basis, a classical LWO requires only small modifications to be reused on the extended topology, and this allows us to reuse all its well-tuned heuristics.

The most innovative key asset of the method is its ability to optimize traffic on intercompartment peering links as well. We have shown on a case study that it does so very efficiently at almost no extra computational cost.

As for a classical LWO, our method can be extended to more general scenarios including several traffic matrices as input and/or possible link failures. Note however that an interTM used as input is likely to be already more stable (and thus representative) than intraTMs. Indeed the intercompartment matrix is invariant under all local early exit fluctuations, e.g. due to failures. This better stability of the intercompartment matrix would allow us to use a smaller set of representative matrices as input, which in turn would give unique link weights settings that are better optimized for each of them.

Even though our method requires additional inputs to build the interTM and some more computation power, this pays off, because our intercompartment-aware LWO clearly outperforms classical (intracompartment) LWO.

A known potential issue with LWOs is route instability. As there is no mutual agreement on the egress/ingress points between ASes, it is not guaranteed that two neighboring ASes (say AS_x and AS_y) running their LWO will not oscillate, one reoptimizing its link weights after the other. Indeed each link weights optimization in AS_x can lead to a change of some egress points, changing the traffic matrix in AS_y which may trigger the reoptimization of the link weights in this AS, and so on, leading to route oscillations.

Such instability may already happen with classical intercompartment-blind LWOs and, as our proposed LWO does not address this issue, some instability may also potentially exist in our case. This is still an open research topic. Instability problems will be addressed in the next chapter.

Finally, this chapter has been accepted for publication [1].

Chapter 3

Self-stability and optimization in presence of multiple autonomous compartments

3.1 Motivation and Objectives

The operational environment enabled by ANA is one of large distributed and decentralized compartments with full autonomy, interacting on behalf of their own objectives and self-interests. This form of networking is highly attractive not only in its self-* properties, but also because it is flexible enough to support diverse and novel networking architectures, protocols and applications. For example, when regions of the networks (e.g. domains, ASes, etc.) are organized into autonomic compartments and publicize themselves as individual transport services, end users (or applications) have the choice of selecting a subset of these compartments to form end-to-end paths [13]. Thus complex user-oriented routing logics (e.g. runtime adaptation, content discovery, etc.) could be executed and QoS assurance could be better supported. In essence, autonomic ANA services are expected to behave according to their own knowledge capability, decision logic, and private goals, thus fostering a large scale distributed environment of independent "entities" sharing a common resource pool (i.e. the networks).

The lack of central coordination in ANA poses a serious question of stability, as evident in some of today's networking problems: BGP routing policy conflicts, overlay vs. underlay route optimization, P2P networking, etc. The key factors that cause this instability in ANA environments are:

- large scale environment of independent “entities"
- local decision making based on self-interest
- concurrency

- dynamic and evolving networking environment
- incomplete and inaccurate information

The individual decisions often lead to sub-optimal system objectives (e.g. load balancing and fair-sharing of resources), and the inevitable resource contention and the resulting self-optimization of ANA compartments may lead to system oscillations and resource deadlocks.

Under the topic of self-stability and optimization in ANA, we strive to find and develop fundamental theories in support of self-stabilization in ANA and analyze (and hopefully suggest) ANA compartment design principles to achieve some degree of global control and resource optimization. To this end, we investigate the concept of self-stability in ANA by establishing realistic analytical models to determine the stability of ANA-like environments and determining methods of convergence; and proposing practical implementation techniques and compartment designs that facilitate self-stability. For experimentation, we will establish the end-to-end QoS-assured path composition and adaptation service as an example demonstration of our findings.

3.2 Concept of Self-Stabilization

The concept of self-stabilization was first introduced in 1973 by Dijkstra [14]. He describes a self-stabilizing system as a distributed system where individual component behavior is determined by a subset of global system state that is known to the component and the global system must exhibit the property of “regardless of its initial state, it is guaranteed to arrive at a legitimate state in a finite number of steps". Two key properties are 1) the system could initiate in any state 2) the system can always recover from transient faults. When discussing self-stabilization, the concept of closure and convergence [15] are often mentioned. The closure property describes that if the system is stable, after a number of executions it will remain stable unless perturbed by external force. The convergence property states that if the system is unstable, it has a tendency to move towards a stable state over time. The length of time that must elapse for this process to terminate is sometimes referred to as the convergence span. It is apparent that self-stabilization exhibits many of the desired self-* properties of self-managing networks and hence is an important area of investigation in autonomic research.

Although there are some existing self-stabilizing algorithms proposed for specific network problems (e.g. the maximum flow problem [16]), thus far few research has been done in studying the feasibility of applying self-stabilization to self-managing networks or analyzing the self-stabilization properties of existing self-management theories. However, we believe this is a promising area of research. For example, many of the proposed theories for the design of self-managing networks exhibit certain degrees of

self-stabilization, such as the dynamic equilibrium concept from game theory, free market principles from Economics, and the emergence concept in biology. We conduct our study using game theory for a number of reasons. 1) Game theory is a well formulated analytical methodology established over the span of decades in economics and mathematics; 2) There exists a large body of work on applying game theory to networking problems and have shown its effectiveness in analyzing various networking scenarios.

In the network context, existing principles of self-stabilization alone are not enough to ensure effective system performance. Two other factors should be considered. 1) we require a measure of “how good" a stable state is. The concept of Price of Anarchy is applicable in that it attempts to quantify the optimality of a stable state. 2) Oscillation among stable states should be avoided. Consider the following scenario: assuming a self-stabilizing network with two legitimate states, each of which has a completely different way of routing traffic through the network. The convergence property states that if the system is in an illegitimate state, it will eventually return to one of the two legitimate states. The closure property assures us that regardless of the time elapsed; the system will stay in legitimate states, unless disrupted by external force. However, these two properties alone are not strong enough to guarantee the stability of a path, since constant oscillation between the two legitimate states is then possible, which results in undesirable route flapping. The general approach to solve this issue is to either create an environment where only one stable state exists, or to ensure once a stable state has been established the system will not seek other stable alternatives.

3.3 Our Investigation on Game Theory

Game theory is a formal analysis methodology applied in environment of self-interested players. Foremost, it answers the question whether there are any stable states in the system. A Nash equilibrium is defined as a system state arrived by considering the strategy choice of all players in the system, where no player has an incentive to deviate from its strategy unilaterally. Thus, a Nash equilibrium represents a stable system state. Ideally, an environment with a unique Nash equilibrium means the system will have a single global stable state and would never deviate from it when the equilibrium is established. However, such games are rare and tend to be quite simplistic. In choosing among its strategies, a player may choose one strategy over the others deterministically or assign a probability for choosing each of its strategies. The Nash equilibrium established by each player deterministically choosing a strategy is called a pure Nash equilibrium, which could be unique. A Nash equilibrium established by each player probabilistically choosing a strategy is called a mixed Nash equilibrium. A mixed Nash equilibrium is never unique and there generally will be oscillation among the player’s strategy choices due to its probabilistic nature. Thus game theory analysis in networking context has largely focused on studying games with pure Nash equilibriums.

Game theoretical analysis has been conducted in many fields of network research in the past (e.g. pricing, flow control, route stability, efficiency of wireless networks, etc.). Some works have examined the existence of unique Nash equilibrium in non-cooperative user-based routing environments [17] [18]. Orda, Rom and Shimkin [17] have shown that in a two-node multi-link network topology, there exists a unique Nash equilibrium. For general networks, the uniqueness of a Nash equilibrium is guaranteed if the cost functions of links have diagonal strict convexity, the users share the same source and destination and are symmetric in cost functions, or each user assigns positive flows to all the links in the network. Altman et al. [18] studied non-cooperative routing games under general topologies with polynomial cost functions. They have shown the uniqueness of a Nash equilibrium under bounded cost. Computation of the Nash equilibrium is carried out as user-based global optimization problem where users have the same source and destination across parallel links and each user assigns positive amount of flow on each links in the network. A special form of routing game termed bottleneck routing game is investigated by Banner and Orda [19]. In such a game, the user attempts to minimize the load of its bottleneck link, rather than to minimize the end-to-end cost. The existence of a Nash equilibrium is shown and for unsplittable flows, a polynomial time convergence bound is obtained.

Of particular interests to us is a special game form called congestion games. This is not only because congestion games are suitable for modeling networks, but also because the existence of a potential means there are effective mechanisms for convergence of the system. Congestion games were first introduced by Rosenthal [20] and later formalized by Monderer and Shapley [21], as a class of games in which the cost of a resource is a non-decreasing function depending on the number of players sharing the resource. A game is called a potential game when there exists a potential function such that the increase in utility of a player (or drop in cost) causes a decrease in potential. A potential game has at least one pure Nash equilibrium. Monderer and Shapley showed that potential games are isomorphic to congestion games. However, Milchtaich [22] showed that this is only the case for unweighed congestion games. A weighed congestion game in which players may have different impacts on the cost function (i.e. varied load among players) may not possess any pure Nash equilibrium. Fabrikant, Papadimitriou and Talwar [23] have shown that the complexity of finding a pure Nash equilibrium in asymmetric congestion games is quite expensive (i.e. PLS-complete), and thus under popular convergence mechanisms (such as the best-reply dynamic), there exist convergence paths of exponential length.

On the topic of convergence to Nash equilibria in congestion games, the single-commodity KP-model [24] is often used. Milchtaich [22] has shown polynomial time convergence exists for players with varied payoff functions. Goldberg [25] bounded the convergence in such games to polynomial time, and Even-Dar and Mansour [26] considered the case in which all players can move simultaneously according to a Nash rerouting policy and have found a polynomial time convergence bound. In multi-commodity congestion games with simultaneous moves, whether convergence to a pure Nash equilibrium could be bounded is still an open question and examples could be found in

which convergence doesn't occur. Because of its complexity, the study of convergence in general congestion games has been mainly focused on finding convergence bound to approximate solutions. Christodoulou, Mirrokni and Sidiropoulos [27] bounded the solution after a discrete round of player moves to O(n)-approximate in general case. Chien and Sinclair [28] showed that when the increase in cost of adding a player is bounded ("bounded jump" condition), convergence to Nash occurs in polynomial time. However in practice, the cost jump by adding a player is often unbounded, and hence it is difficult to provide a bounded approximation because of the possibility that a player with low payoff gain is unwilling to move and thus "locks out" another player's chance of obtaining a high payoff gain. Goemans, Mirrokni and Vetta [29] studied convergence of Nash dynamics to "sink equilibrium", which is not an approximate of a pure Nash equilibrium. In fact, a sink equilibrium could be formed by a group of cyclic states in some cases. In bounding the optimality of an equilibrium, Price of Anarchy (PoA) is frequently used. It defines the ratio between the social cost of an equilibrium and the social optimal. Some research has been conducted on bounding the PoA ratio in congestion games and in some cases tight bounds are found.

3.4 Concept of Desirability

As a starting point, we propose and examine a variation to the traditional Nash equilibrium. We define desirable equilibrium as a Nash equilibrium where all resources in the system are not in "congestive state". We consider a congestive state of a resource as the level of work load under which the resource can no longer provide satisfactory performance. In essence, our definition of desirability lends itself to study the problem of distributed load balancing under which the maximum load to capacity ration on any resource in the system is bounded. This is in effect a special case of Price of Anarchy where the social cost being considered is the maximum load to capacity ratio of the system.

The traditional definition of potential in congestion games does not guarantee desirability of a Nash equilibrium. Figure 3.1 shows such an example of a running congestion game. There are 8 players with an identical strategy set, 6 of the players are in equilibrium with respect to each other as depicted in the graph. The table shows the payoffs of the two remaining players under different strategy profiles. The bold numbers are the computed potential of the system under the specific profile. A '*' symbol besides a payoff indicates it is the player's best-response to the other player's particular strategy. Thus, if a state has two '*', then it is a Nash equilibrium. We see that the state with the lowest potential is an equilibrium between the two players, and in fact is a pure Nash equilibrium for this 8-player game, as we would expect. However we observe that under this equilibrium state, resource t_5 is in an overloaded condition, yet no player is willing to play another strategy unilaterally. We argue that such a state is a bad configuration for the overall system as the overloaded resource is adversely affecting all players that use it.

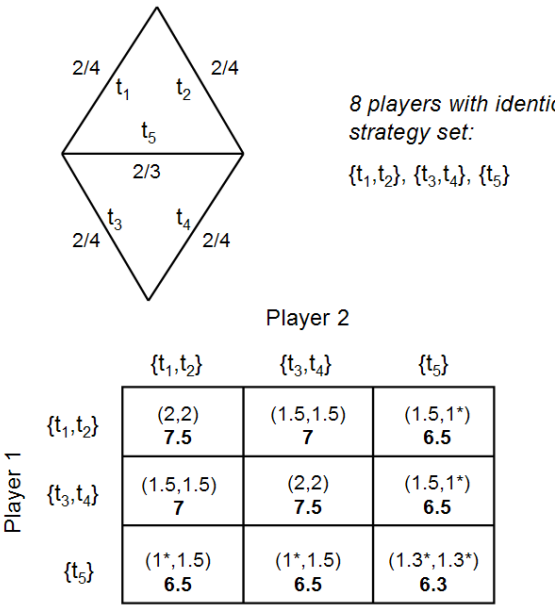


Figure 3.1: A Running Example of A Congestion Game with Potential

Thus, as a first step, we aim to define a congestion game model that can incorporate realistic system settings and also provide identifiable desirable equilibriums. Ideally, the game should only produce equilibriums that are desirable. Then formal analysis needs to be conducted on studying the existence of desirable equilibriums in such games and the theoretical bound on converging to such desirable equilibriums. Our preliminary simulation results on such games suggest the existence of desirable equilibriums is likely and on average convergence could be obtained in polynomial time. However, to realize such games in practice, implementation techniques need to be investigated and designed. All of these objectives are part of our ongoing work in ANA for the coming years.

Chapter 4

Overlay compartment topology optimization over mobile underlay compartments

4.1 Introduction

The overlay technique consists in building a virtual topology over the physical topology of a network. In ANA this means laying a compartment over another one. The former is the overlay compartment, and the latter is the underlay compartment. Overlays have been successfully proposed in the Internet for improving the basic routing service provided by IP for a variety of applications. They can afford for example a multicast functionality, object location, secure data dissemination, content-distribution, quality of service or fault-tolerant routing. They have also been successful for in-situ testing of new networking solutions, a field that recently attracted a growing interest [30].

Many overlay applications have been proposed, and many with their own proprietary functionality support. Several overlay functionalities are covered by different works. These are for example the overlay topology discovery, routing path selection, fault detection and tolerance, overlay link performance estimation or resource allocation. Hence, the idea of designing a general unified framework for various application-specific overlays emerged [31–33]. A generic overlay solution allows the designer of an overlay application to concentrate on its application goal and needs rather than on the maintenance of the overlay.

There are also several incentives for deploying overlays over Mobile Ad Hoc Networks (MANETs). Firstly, the ad hoc users are likely to desire the same services as the one offered when they are connected through wired or infrastructure-based wireless access to the Internet. Secondly, ad hoc networks and overlay applications are faced with the same fundamental challenge of providing connectivity in a decentralised, dynamic environment. In both domains, the user devices work simultaneously as end-system and routing nodes. This is an advantage compared to the Internet, where core routers that

take an active role in an overlay cannot benefit from useful information own by the user application, and where end-host based overlays suffer from performance limitations. However, the maintenance of an overlay on top of an ad hoc network is challenging because of mobility and low bandwidth.

In a wired context, a complete decoupling of overlay topology and underlying data network topology may not be completely desirable, because it leads to inefficiencies. This is even more so in ad hoc networks, where the maintenance of many long tunnels is not even conceivable, as mobility makes long routes very unreliable. The cost of rebuilding long tunnels may be important, in terms of delay and of bandwidth. Hence, the benefits of the overlay application could be completely lost if overlay neighbours were not close to each other.

The Overlay Topology Control protocol (OTC), briefly presented in this chapter, builds and maintains a locality-aware overlay on top of a mobile ad hoc network. We do not specify its application domain and evaluate the virtual structure obtained by an indirect performance evaluation based on overlay flooding. The diffusion of message can be seen as a worst-case scenario for group communication. It is also a key component of many unicast route discovery mechanisms in MANETs.

4.2 Related work

The deployment of overlays in MANETs has not yet received a lot of attention. In [34], an overlay approach is proposed for service discovery in ad hoc networks. A new layer is inserted above the transport layer in order to build, maintain, and offer an overlay structure that optimally serves the mechanisms of service trading and efficiently adapts to the changing physical topology. The maintenance protocol consists of an adaptation of CAN that lightens its structure. It selects short overlay links and reduces the control traffic. The adaptation of application-layer multicast to the ad hoc environment was proposed by the AMRoute protocol [35], which establishes a mesh between the multicast users and then a tree for data distribution over the mesh. More proposals are available for P2P content distribution in MANETs. Many of them exploit cross-layer interactions with the routing protocol [36–38], or the node position knowledge [39] or even introduce the necessary p2p functions at the network routing layer [40, 41]. An elegant p2p design for ad hoc networks is proposed in [42]. However, the authors do not take into account mobility as they are mostly interested in evaluating the feasibility of DHT lookup in ad-hoc networks.

In all above cited works, the use of logical long-range neighbours is avoided because of their prohibitive maintenance cost. Another common point is that the overlay quality is evaluated by the level of performance obtained by its user application.

4.3 The Overlay Topology Control protocol

4.3.1 Assumptions and protocol overview

We consider a connected ad hoc network, the underlay compartment. The underlay routing protocol is supposed to provide short paths, but not necessarily the shortest ones and may build asymmetric paths. We do not assume that the underlay routing protocol is able to inform the above layer about the length of the available paths. We make however the following, weaker, assumption: When a node receives a packet, it is able to know how many hops the packet has traversed since its emission.

The OTC algorithm is fully distributed and local, i.e. each overlay node exchanges only a few messages with a limited number of nearest overlay nodes. It avoids logical long-range neighbours because of their prohibitive maintenance cost. The overlay topologies built are connected, at least with a high probability.

We differentiate broadcast and unicast overlay neighbours. The former are also physical neighbours, i.e. there exists a direct radio communication link between them, while the distance between the latter is at least of two hops.

Each overlay node U:

1. Collects in its neighbour list L_U the identifier and the shortest distance to its K closest overlay nodes.
2. Also inserts in L_U any overlay node V such that $U \in L_V$ (thus turning the neighbourhood relation into a symmetric relation).
3. Selects its active neighbours in L_U by applying the pruning rule described below (in Sec. 4.3.3).
4. For resilience, sets all broadcast neighbours as active, and does not prune the overlay links established with the 3 nearest overlay nodes.

Let us define the overlay density as the proportion of overlay members. We observed in an extensive set of simulations that setting K to the value of 8 was sufficient to guarantee connectivity of overlay topologies with a probability higher than 95% for up to 1000 underlay nodes and overlay densities ranging from 10 to 90% [2].

The data messages only flow onto active neighbours, i.e. on active overlay links. Oppositely, OTC control messages are continuously exchanged with pruned neighbours as well as with selected ones.

4.3.2 Neighbours list maintenance

Broadcast neighbours are discovered by *otc_hello* messages, encapsulated in broadcast packets with the Time To Live field (TTL) set to one. Unicast neighbours are discov-

ered with an increasing ring search method, using *otc_request* messages, and *otc_reply* messages.

When an overlay node U inserts a node V in its neighbours list, the distance associated to V equals the number of hops h that the reply from V has traversed. For any overlay link, one and only one overlay node is responsible for monitoring its length. The other end node sends to it an *otc_advertise* at regular intervals. By observing the number of hops traversed by the advertisements, the monitoring node is able to detect any overlay link length modification.

Advertisements are also sent asynchronously anytime an overlay link is modified. Their purpose is to give to the two end nodes a consistent view of the (local) overlay topology, so as to make their decisions consistent. They contain the local neighbours list, sorted by increasing distance and, when distances are equal, by increasing identifier. Modification events are the creation of the overlay link, and update of its length or state. An overlay link may be in the active or pruned state (see next section).

Deletion of an overlay link may occur when its two end nodes move away from each other. It is announced by the emission of an *otc_delete* message.

4.3.3 Active neighbours selection

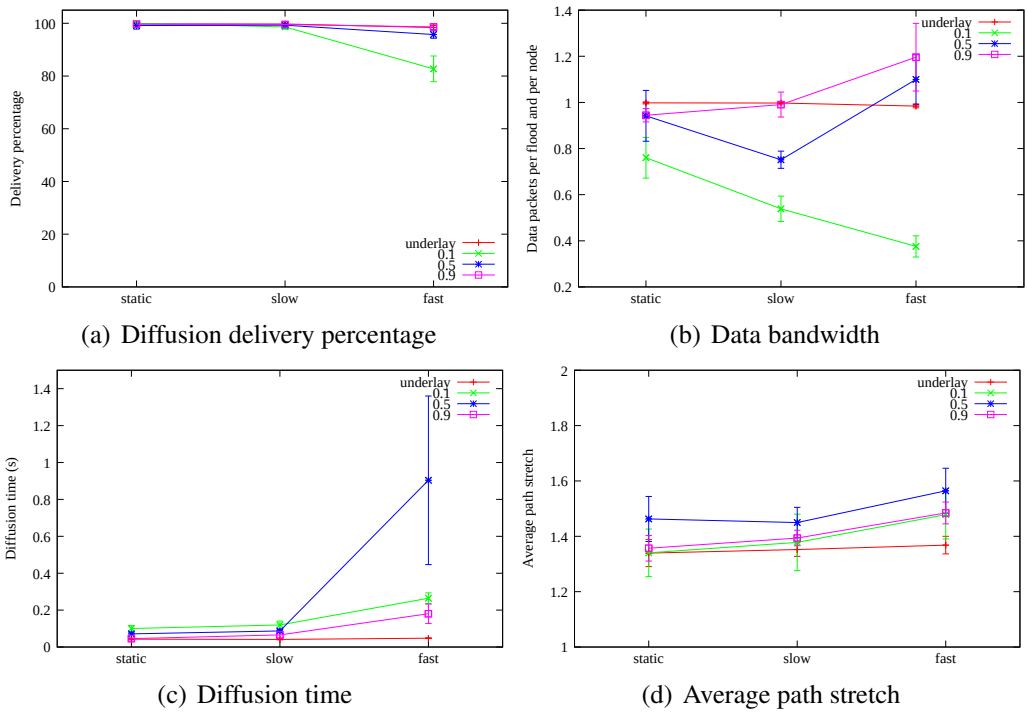
Consider an overlay link (U, V), with U elected as the monitoring node. Node U receives at regular intervals an advertisement that contains L_V. It prunes the overlay link (U, V) if and only if there exists a third overlay node W that appears before V in L_U and before U in L_V, and such that $d(U, W) + d(W, V) \leq \alpha d(U, V)$. Parameter α is called the *pruning selectivity*. It is a real number in the interval $(1, 2)$.

The idea is, for every pair of overlay neighbours (U, V) to prune each other if they share a common better neighbour, under the condition that the remaining overlay path that links them is not stretched by more than a factor α[1]. The pruning rule does not affect the overlay connectivity. It reduces significantly the bandwidth consumption during flooding and decreases contention, while increasing the overlay path stretch and diffusion time by an acceptable amount.

4.4 Evaluation

To assess our OTC protocol we have chosen to run it over a particular MANET underlay compartment, namely the Optimized Link State Routing protocol (OLSR) [43]. As OTC is not application-specific, its evaluation uses indirect performance criteria based on overlay flooding. The diffusion of message can be seen as a worst-case scenario for group communication. It is also a key component of many unicast route discovery

[1]Note however that the final maximal overlay stretch may be higher than α because the pruning rule is also applied to (U, W) and (W, V), and so on.



(a) Diffusion delivery percentage

(b) Data bandwidth

(c) Diffusion time

(d) Average path stretch

Figure 4.1: Overlay flooding performance on OTC topologies built over OLSR

mechanisms in MANETs.

We use the ns-2.29 simulator and its random waypoint (RWP) scenarios generator. Parameters used are given in Tab. 4.1. The OTC protocol starts at the beginning of the simulation, with a pruning selectivity set to 1.5. After 30 seconds, a source overlay node starts to emit 100 broadcast overlay messages of 64 bytes, at the average rate of one message per second.

Topologies		**Scenarios**		
Number of nodes	100	**RWP**	Speed (m/s)	Pause Time (s)
Field length	1200 m	Static	0	150
Communication range	250 m	Slow	$(1, 5)$	$(0, 10)$
Overlay density	0.1, 0.5, 0.9	Fast	$(5, 15)$	$(0, 5)$

Table 4.1: Simulations parameters

Performance obtained over OLSR are shown on Fig. 4.1. The 95%-confidence intervals are indicated. As a reference, we also evaluated the diffusion of a broadcast packet on the underlay.

Diffusion delivery percentage For all overlay nodes, except the overlay broadcast messages source, we determined how many overlay messages out of the hundred sent were received before the end of the simulation.

The diffusion delivery percentage over OLSR is very good in the static and slow case, for all overlay densities. When nodes move a lot and rapidly, the diffusion delivery percentage is also very good (above 95%), except for the lowest overlay density (only 83% of the overlay nodes receive the flooded message).

Bandwidth consumption Fig. 4.1(b) shows the number of data packets emitted per flood and per ad hoc node. In the static and slow cases, or when the overlay density is very low, the flooding of a message on the overlay, which is able to propagate useful information for overlay routing, costs less than one packet per node. We also observed that the total control bandwidth, composed of OTC and OLSR packets, has the OLSR traffic as major component. The OTC traffic represents less than 40% of the control packets in all cases, less than 20% at the lowest overlay density.

Diffusion time For each overlay message flooded on the overlay, we logged the interval of time elapsed between its emission and the moment at which its first copy was received by the last overlay node.

The overlay diffusion time raises with the mobility degree. The sharper increase is observed at the middle overlay density because each overlay node must emit a new unicast packet for each of its unicast neighbours. At a high overlay density, most neighbours are reached by a single, on-hop, broadcast packet. At low density, there are a few overlay links, thus the time required for accessing the media is lower.

Overlay path stretch Each time an overlay message was received, we also computed the ratio of the number of hops it has passed through since its emission and of the shortest path length from the source. This defines the path stretch. Its average value on the overlay, shown on Fig. 4.1(d), is under 1.6 for all cases studied. Note that the average underlay path stretch is also greater than 1.2 because of collisions. We also observed, at medium and high overlay densities, that the maximum path stretch on the overlay and on the underlay are almost equal. At the lowest overlay density, the maximum path stretch on the overlay is 20% under its value on the underlay.

4.5 Conclusions

While service overlays have proved their utility in the Internet, there are currently only a few proposals in the context of ad hoc networks.

The OTC protocol builds and maintains a generic service overlay over a mobile ad hoc network. It is not application-specific, avoids as much as possible cross-layer optimisations and assumptions about the underlay topology or routing protocol.

The presented evaluation utilises performance criteria based on overlay flooding. The diffusion of a message on overlay topologies built by OTC always shows up good

performance, except when the overlay density is very low and the mobility degree high. Over OLSR the overlay maintenance generates an acceptable volume of control traffic.

An extended version of this chapter, with the complete description of OTC, and a deeper evaluation, has been for publication [3].

Chapter 5

Data Placement in an Overlay Compartment to Minimize Inter-Compartmental Traffic

5.1 Introduction

The challenge of placing data items in a network so that requests can be efficiently and reliably satisfied is a broad and well studied problem [44–46]. The applications range over several domains such as distributed databases, parallel computing, facility location and sensor networks. Moreover, data placement is a critical aspect not only in static environments, but also in the context of dynamic settings and autonomic networks [47, 48]. Here, we focus on this issue from the perspective of data placement in a peer-to-peer (P2P) network under the ANA framework. The traffic generated by P2P networks has become a topical subject in recent years. The success of applications such as Bit-Torrent, eDonkey, Kazaa and others have fostered a burgeoning community engaged in online file-sharing. This phenomenon has been a boon to Internet-Service Providers (ISPs) who have witnessed increased revenues generated by a swelling customer base. For instance, many ISPs now offer tiered bandwidth packages aimed at attracting those customers willing to pay for increased bandwidth [49]. However, the rise of P2P applications also poses a new challenge for ISPs whose respective infrastructures are now facing increased and sustained traffic loads [50]. While the exact volume of total traffic generated by P2P applications is disputed there is a general consensus that such traffic constitutes a significant percentage of all network traffic with reported values as high as 80% [49,50]. Consequently, other customers can suffer due to bandwidth starvation; moreover, there is a spill-over effect where traffic passing through a heavily loaded domain is also affected. Such behavior can also have economic impact on ISPs since a significant percentage of P2P traffic has been reported to traverse peering links between ISPs [51]; such inter-AS traffic is particularly expensive.

These problems are not necessarily due to the number or size of files being shared between peers, but rather result from topological discrepancies between the overlay and

underlay networks. This situation between P2P users and Internet entities such as ISPs is likely to continue posing difficulties for future network designs such as ANA. Clearly, P2P traffic between ANA compartments should be minimized so as utilize available bandwidth effectively. However, there is no easy solution to this problem in the flexible and completely decentralized architecture of the proposed ANA architecture. As economic motivation will still likely be a factor of future Internet designs, it seems likely that entities such as ISPs will still play a vital role. In this document, we seek to address how such entities can cooperate with P2P users in shaping of P2P traffic. For simplicity, we will refer to such entities as ISPs for the remainder of this document.

In terms of the ANA architecture, routing occurs both within and between compartments in the underlay. Inter-compartmental routing will no doubt incorporate contractual agreements and compartmental routing policy. In sharp contrast to this, routing in overlay networks is far more flexible. Links between peers are largely unconstrained and routing can be accomplished by a number of common techniques ranging from flooding to the use of distributed hash tables [52]. Generally, requests for data amongst peers can be resolved with a small hop-count and small path-stretch; this is an issue that has received much attention from the academic community [52–54]. Unfortunately, the desire of peers to select for optimal paths is often in conflict with the aims of ISPs who seek to optimize the performance of their compartment as a whole.

In today's internet, ISPs have few options at their disposal when confronted with this problem. Caching has been utilized by many ISPs and several companies such as CacheLogic, PeerApp, Tangium and others offer a number of solutions [50]. The legality of caching has recently been the subject of intense scrutiny since the data items themselves are stored by the ISP which, it has been suggested, potentially violates copyright. Recent legal precedent would seem to validate the use of caching by Canadian ISPs [55]; however, the issue is less clear in other countries and, in these cases, resolving the legality of caching may involve protracted legal procedures [56]. Another technique regularly employed by ISPs is traffic engineering (TE). A simple, but common, approach is to throttle or rate-limit the traffic of a particular user once a bandwidth threshold has been exceeded; such throttling may also be accompanied by threats of legal repercussions in the hopes of deterring such behavior in the future. While this approach to TE achieves a reduction in traffic, it can also induce clientele to switch to less restrictive ISPs [50]. Even worse, packet encryption and the sending of large data items through the network, rather than from holder to requester, may be employed by peers to enhance privacy and improve the anonymity of the data holder; this response to such "solutions" only exacerbates the problem.

It is apparent that the current situation is far from optimal for both ISPs and users of P2P applications. In this work, we propose a cooperative approach to aid ISPs in reducing inter-compartmental traffic which, due to peering agreements in future Internet designs, is likely to be a more significant concern than intra-compartmental traffic. Our work differs from previous approaches by having ISPs optimize the placement of data in the network so as to bias traffic. We do not require ISPs to hold data nor do we explicitly interfere with the overlay topology. This removes the need for the acquisition of expensive equipment necessary for caching solutions and may aid in avoiding issues of copyright violation.

5.2 Related Work

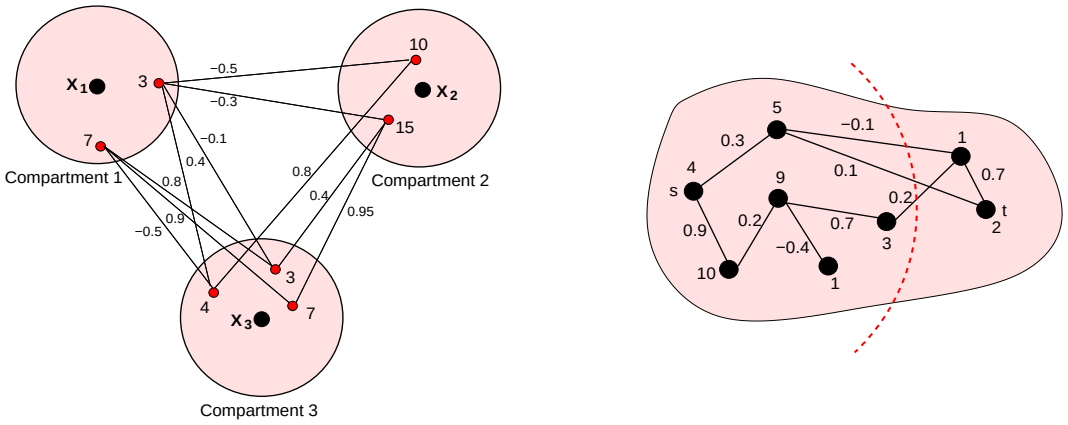
The fundamental mismatch between overlay and underlay topologies has been the subject of much study. Several works have proposed incorporating underlay information on geographical locality when forming and utilizing P2P network connections [57, 58]. Experimental results have shown that such approaches can indeed reduce bandwidth usage and result in more efficient handling of data requests. Along similar lines, semantic overlays networks (SONs) attempt to leverage content locality in order to achieve better performance [59–61]. However, these techniques are primarily concerned with optimization from the perspective of the peers with any resulting benefits to the ISP being secondary; hence, they do not directly address the core problem.

A point of concern is the need for peers to be aware of the underlay topology so as to behave in an informed fashion. However, such knowledge can lead to unstable situations where modifications to the overlay provoke changes in the underlay and so forth in a potentially perpetual cycle. Recent work by Aggarwal *et al.* [62] proposes an elegant solution whereby an oracle service aids in the process of placing new peers into the overlay. This oracle service is informed by the ISPs themselves, thus avoiding possible feedback complications. The authors show that by using the oracle to bias the location of join events into the network yields benefits to both ISPs and P2P users. However, there are some drawbacks to this approach. First, the bias introduced by the oracle when forming the network may result in a less robust architecture due to reduced connectivity. Second, it is unclear what benefits such an oracle yields to structured P2P systems since the ISP may have little interest in maintaining the architecture necessary to maintain performance guarantees. In [51], the authors propose tricking ISP caching proxies into storing P2P traffic data in order to relieve network congestion. This is shown to provide performance improvements; however, it relies on being able to store data efficiently on the caching proxies which is not always possible due to the lack of control over such proxies.

5.2.1 Unbalanced Cuts and Semidefinite Programming

Since the seminal work of Goemans and Williamson [63], semidefinite programming (SDP) has figured prominently in the design of improved approximation algorithms for many NP-complete problems. In addition to designing approximation algorithms, SDP has been employed extensively in the development of heuristics for combinatorial problems in the areas of phylogenic reconstruction [64], machine learning [65], sensor network layout [66], bioinformatics [67] and graph partitioning [68]. These results have demonstrated that heuristics can benefit greatly from this optimization technique. Unbalanced graph cuts have recently received attention due to their usefulness in modelling problems from the domains of data placement, community finding and epidemic containment [69, 70].

For our purposes, we are concerned with a problem MIN-MAX MULTIWAY CUT developed by Svitkina and Tardos [5] which models many issues of P2P traffic. As input to the problem, we are given an undirected graph $G = (V, E)$ with edge capacities, node



(a) (b)

Figure 5.1: (a) An illustration of the MIN-MAX MULTIWAY CUT problem where each edge is weighted with a correlation value between nodes which represent data items in the network. (b) An illustration of the MAXSBCC problem.

weights and a set of terminals $X = \{x_1, ..., x_k\} \subseteq V$. The output is a partitioning of G into sets $S1, ..., S_k$ such that $V = \bigcup S_i$ and $x_j \in S_j$ for $j = 1, ..., k$. Furthermore, if we let $\delta(S_i)$ represent the capacity of the cut between S_i and $\bigcup_{j \neq i} S_j$, we want a partitioning of G that minimizes the maximum $\delta(S_i)$ over all i. Figure 5.2.1(a) depicts a simple instance of the MIN-MAX MULTIWAY CUT. One way of solving this problem is to repeatedly solve a smaller subproblem known as MAXIMUM-SIZE BOUNDED-CAPACITY (MaxSBCC). Given an undirected graph $G = (V, E)$ with weights on the vertices $w(v)$, capacities on the edges $c(e)$, source and sink vertices $v_s, v_t \in V$, and an integer B, the MAXSBCC problem is to find an s-t cut (S, T) such that $\delta(S) \leq B$, and $w(S) = \sum_{v \in S} w(v)$ is maximized; Figure 5.2.1(b) illustrates this problem. A (α, γ)-bicriteria approximation algorithm for MAXSBCC is an algorithm that, given an instance of MAXSBCC with an optimal solution (S^*, T^*), returns in polynomial time a solution (S', T') such that $\delta(S') \leq \alpha B$ and $w(S') \geq \gamma w(S^*)$ where $\alpha \geq 1$ and $0 < \gamma \leq 1$. Current approaches to this problem allow only an approximation result for capacity of the cut and not the size of the source partition which limits its flexibility. Our current work addresses this issue by designing an approximation algorithm for MAXSBCC that allows for a trade-off in performance on both the capacity of the cut and source size.

5.3 Current Results and Discussion

The MIN-MAX MULTIWAY CUT allows us to formulate an initial model of inter- compartmental traffic. Each terminal x_i can be viewed as a compartment to which vertices, representing data items, can be assigned. When a data item is assigned to a terminal, it is assumed to be stored by a peer in the overlay network that inhabits that compartment. The edge capacities correspond to correlation values between data items as measured over time by the ISP. For instance, if files are broken into segments, a search on one segment may induce a subsequent search for the next segment and so forth. Since the

MIN-MAX MULTIWAY CUT problem minimizes the maximum cut, optimizing over this problem evenly distributes inter-compartment traffic over all compartmental peering relationships. Using semidefinite programming, we have developed a heuristic for MAXSBCC subproblem on graphs with total positive correlation value (ie. the sum of all edges is positive).

Given this optimization problem, ISPs can use the correlation data they extract from their compartment and, in cooperation with other ISPs, obtain a solution which suggests more effective data placement schemes for the current P2P system. The exact protocol by which peers and ISPs might come to agreement over any particular data assignments is still under discussion. Initial experiments with MAXSBCC have yielded promising results. Currently, we are looking into scaling up our experiments using data collected on actual P2P networks. This data provides full information on the item being offered by each peer as well as a record of time-stamped file transfers in the network which allows for the calculation of correlation values. Incorporating this data into a meaningful topology such as the AS-graph is currently underway. Modifications to the formulation to take into account replication and dynamism are also being examined.

Chapter 6

Conclusion

This deliverable is a first step in the direction of addressing self-optimization in the ANA architecture. We have described optimization issues encountered in large distributed systems, when every compartment can run its own optimization algorithm. Firstly we have considered compartments forming a structured but flat topology, and we have shown how a compartment may reach a better optimum if it is aware of the intercompartment level. We have also addressed possible instabilities resulting from the impact a compartment optimization may have on others. Secondly we have considered compartments laid over one another, and have shown how an overlay compartment can optimize its topology with no cooperation from the underlay compartment. We have also addressed this kind of multilayer optimization using a cooperative approach to optimize the placement of data at the overlay level.

During the next 18 months, besides pursuing the research topics initiated in chapters 3 and 5, we will address new topics that emerged in relation with the monitoring activities. Firstly, the Multi-Compartment Information Sharing (MCIS) that has been designed to store, index and search the monitoring data (see Deliverable D3.3) is currently not self-adaptive. We will find out solutions for adaptive indexing of the range data as opposed to the systematic static indexing approach employed currently by the system. Our objective is to find indexing strategies that adapt the indexing process based on the usage (e.g. query and storage rates and data properties) and the environment (e.g. number of nodes). Currently, the MCIS system applies explicit load balancing strategy for each data compartment separately in order to cope with unevenly distributed values of data. Our objective is to extend the load balancing strategy on another level, namely across these data compartments, i.e. to come up with multi-compartment load balancing strategies. Currently the MCIS supports only single attribute indexes via the attribute hubs. We aim to evaluate how we can extend the system to support multi-attribute indexes.

Secondly, in deliverable D3.3, we have also shown how a coordinate system can be used in the monitoring framework to optimize the RTT measurements, but it is known that such coordinate systems are not accurate when TIVs (Triangular Inequality Violations) are present in the network. We proved that a hierarchical coordinate system, running both a local and a global coordinate systems, with the goal of having very few

TIVs at the local level, would be more accurate. The next step consists in designing an algorithm that will allow nodes to cluster themselves autonomically to form a two-tier and accurate coordinate system.

Bibliography

[1] S. Balon and G. Leduc. Combined intra- and inter-domain traffic engineering using hot-potato aware link weights optimization. In *Proc. of ACM SIGMETRICS - International Conference on Measurement and Modeling of Computer Systems, Annapolis, Maryland, USA*, 2008.

[2] S. Calomme and G. Leduc. Efficient and resilient overlay topologies over ad hoc networks. In *Proc. 2nd International Workshop on Self-Organizing Systems (IWSOS 2007), UK*, 2007.

[3] S. Calomme and G. Leduc. An overlay maintenance protocol for overlay routing on top of ad hoc networks. In *Proc. of IFIP Networking 2008, Singapore*, 2008.

[4] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Proceedings of INFOCOM*, pages 519–528, 2000.

[5] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS Weights in a Changing World. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[6] B. Fortz and M. Thorup. Robust optimization of OSPF/IS-IS weights. In *Proceedings of INOC*, pages 225–230, October 2003.

[7] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, pages 265–279, June 2001.

[8] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, and Jennifer Rexford. NetScope: Traffic engineering for IP networks. *IEEE Network Magazine*, pages 11–19, March/April 2000.

[9] Jennifer Rexford. *Handbook of Optimization in Telecommunications*, chapter Route optimization in IP networks. Springer Science + Business Media, February 2006.

[10] Selin Cerav-Erbas, Olivier Delcourt, Bernard Fortz, and Bruno Quoitin. The Interaction of IGP Weight Optimization with BGP. In *Proceedings of ICISP*, Cap Esterel, France, August 2006.

[11] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. Guidelines for interdomain traffic engineering. *ACM SIGCOMM Computer Communications Review*, October 2003.

[12] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1):83–86, January 2006.

[13] J. Xiao and R. Boutaba. QoS-aware service composition and adaptation in autonomic communication. *IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Autonomic Communication Systems*, 23(12):2344–2360, 2005.

[14] E.W. Dijkstra. Self-stabilization in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

[15] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25(1):45–67, March 1993.

[16] S. Ghosh, A. Gupta, and S. Pemmaraju. A self-stabilizing algorithm for the maximum flow problem. In *IEEE 14th Annual International Phoenix Conference on Computers and Communications*, pages 8–14, 1995.

[17] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on Networking*, 1(5):510–522, October 1993.

[18] E. Altman, T. Basar, T. Jimenez, and N. Shimkin. Competitive routing in networks with polynomial cost. In *IEEE INFOCOM 2000*, volume 1586-1593, 2000.

[19] R. Banner and A. Orda. Bottleneck routing games in communication networks. In *IEEE INFOCOM 2006*, April 2006.

[20] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[21] D. Monderer and L. Shapley. Potential games. *Games and Economics Behavior*, 14:124–143, 1996.

[22] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.

[23] A. Fabrikant, C.H. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *36th ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004.

[24] E. Koutsoupias and C.H. Papadimitriou. Worst-case equilibria. In *Symposium on Theoretical Aspects in Computer Science (STACS)*, 1999.

[25] P.W. Goldberg. Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In *25th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 131–140, 2004.

[26] E. Even-Dar and Y. Mansour. Fast convergence of selfish rerouting. In *16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 772–781, 2005.

[27] G. Christodoulou, V.S. Mirrokni, and A. Sidiropoulos. Convergence and approximation in potential games. In *Symposium on Theoretical Aspects in Computer Science (STACS)*, 2006.

[28] S. Chien and A. Sinclair. Convergence to approximate Nash equilibria in congestion games. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[29] M. Goemans, V. Mirrokni, and A. Vetta. Sink equilibria and convergence. In *46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 142–154, 2005.

[30] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: realistic and controlled network experimentation. In *Proc. of 2006 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pisa, Italy, September 2006.

[31] Z. Li and P. Mohapatra. Qron: Qos-aware routing in overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[32] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. Overqos: offering internet qos using overlays. *SIGCOMM Comput. Commun. Rev.*, 33(1), 2003.

[33] J. Touch. Dynamic internet overlay deployment and management using the x-bone. *Computer Networks*, 36(2-3), July 2001.

[34] M. Klein, B. Konig-Ries, and P. Obreiter. A lightweight overlay for service discovery in mobile ad hoc networks. In *Proc. of 3rd Workshop on Applications and Services in Wireless Networks (ASWN'03)*, Berne, Switzerland, July 2003.

[35] J. Xie, R. R. Talpade, A. McAuley, and M. Liu. Amroute: Ad hoc multicast routing protocol. *Mobile Networks and Applications*, 7(6), December 2002.

[36] M. Conti, E. Gregori, and G. Turi. A cross-layer optimization of gnutella for mobile ad hoc networks. In *Proc. of ACM MobiHoc'05*, pages 343–354, 2005.

[37] F. Delmastro. From pastry to crossroad: Cross-layer ring overlay for ad hoc networks. In *Proc. of 2nd IEEE International Workshop on Mobile Peer-to-Peer Computing (MP2P'05)*, Kauai Island, HI, March 2005.

[38] H. Pucha, S.M. Das, and Y.C. Hu. How to implement dhts in mobile ad hoc networks? In *Proc. 10th ACM International Conference on Mobile Computing and Network (MobiCom'04)*, Philadelphia, PA, September 2004.

[39] C. Cramer and T. Fuhrmann. Proximity neighbor selection for a dht in wireless multi-hop networks. In *Proc. 5th IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, Konstanz, Germany, August 2005.

[40] T. Zahn and J.H. Schiller. Madpastry: A dht substrate for practicably sized manets. In *Proc. 5th Workshop on Applications and Services in Wireless Networks (ASWN'05)*, Paris, France, June 2005.

[41] H. Pucha, S.M. Das, and Y.C. Hu. Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks. In *Proc. 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, English Lake District, UK, December 2004.

[42] R. Kummer, P. Kropf, and P. Felber. Distributed lookup in structured peer-to-peer ad-hoc networks. In *Proc. of International Conference on Distributed Objects and Applications (DOA'06)*, Montpellier, France, October 2006.

[43] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr).

[44] Bo Sheng, Qun Li, and Weizhen Mao. Data storage placement in sensor networks. In *The ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006.

[45] Leana Golubchik, Sanjeev Khanna, Samir Khuller, Ramakrishna Thurimella, and An Zhu. Approximation algorithms for data placement on parallel disks. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2000.

[46] Sudipto Guha and Kamesh Munagala. Improved algorithms for the data placement problem. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

[47] Nikolaos Laoutaris, Georgios Smaragdakis, Konstantinos Oikonomou, Ioannis Stavrakakis, and Azer Bestavros. Distributed placement of service facilities in large-scale networks. In *INFOCOM 2007,* 26^{th} *IEEE International Conference on Computer Communications*, 2007.

[48] Konstantinos Oikonomou and Ioannis Stavrakakis. Scalable service migration: The tree topology case. In *IFIP Fifth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2006)*, 2006.

[49] Thomas Mennecke. DSL broadband providers perform balancing act, `www.slyck.com/news.php?story=973`.

[50] Peter Heywood. Caching in on P2P, `www.lightreading.com/document.asp?doc_id=34399`.

[51] Guobin Shen, Ye Wang, Yongqiang Xiong, Ben Y. Zhao, and Zhi-Li Zhang. HPTP: Relieving the tension between ISPs and P2P. In *Proceedings of International Workshop on Peer-To-Peer Systems (IPTPS)*, 2007.

[52] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[53] Ittai Abraham, Dahlia Malkhi, and Oren Dobzinski. Land: Stretch $(1+\epsilon)$ locality-aware networks for DHTs. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[54] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001.

[55] Javad Heydary and Christopher Zaleski. In landmark decision (Tariff 22), supreme court rules ISPs not liable for copyright violations by subscribers, `www.heydary.com/publications/copyright-law-canada.htm`.

[56] David Canton. Website caching still a thorny issue, `www.canton.elegal.ca/archives/2006/03/website_caching.html`.

[57] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM 2006.IEEE Communications Society*, 2006.

[58] Micah Adler, Rakesh Kumary, Keith Ross, Dan Rubensteinx, Torsten Suel, and David Yao. Optimal peer selection for P2P downloading and streaming. In *Proceedings of IEEE INFOCOM 2006. IEEE Communications Society*, 2006.

[59] Chunqiang Tang, Zhichen Xu, and Mallik Mahalingam. pSearch: Information retrieval in structured overlays. In *ACM SIGCOMM Computer Communication Review*, 2003.

[60] Shu-Ming Shi Zheng Zhang and Jing Zhu. SOMO: Self-organized metadata overlay for resource management in P2P DHT. In *Proceedings of International Workshop on Peer-To-Peer Systems (IPTPS)*, 2003.

[61] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim Van Pelt. Gridvine: Building internet-scale semantic overlay networks. In *International Semantic Web Conference (ISWC)*, 2004.

[62] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can ISPs and P2P users cooperate for improved performance. In *ACM SIGCOMM Computer Communication Review*, 2007.

[63] Michel Goemans and David Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115 – 1145, 1995.

[64] Shlomo Moran, Satish Rao, and Sagi Snir. Using semi-definite programming to enhance supertree resolvability. In *Proceedings of the* 5^{th} *Workshop on Algorithms in Bioinformatics*, 2005.

[65] Yi Zhang, Samuel Burer, and W. Nick Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7(December):1315–1338, 2006.

[66] Michael Carter, Holly Jin, Michael Saunders, and Yinyu Ye. SPASELOC: An adaptive subproblem algorithm for scalable wireless sensor network localization. *SIAM Journal on Optimization*, 17(4):1102–1128, 2006.

[67] Konstantinos Kalpakis and Parag Namjoshi. Haplotype phasing using semidefinite programming. In *Proceedings of the* $5th$ *IEEE Symposium on Bioinformatics and Bioengineering*, 2005.

[68] Bissan Ghaddar, Miguel Anjos, and Frauke Liers. A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem, 2007.

[69] Zoya Svitkina and Éva Tardos. Min-max multiway cut. In *Proceedings of RANDOM-APPROX*, 2004.

[70] Ara Hayrapetyan, David Kempe, Martin Pál, and Zoya Svitkina. Unbalanced graph cuts. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA)*, 2005.