Challenges in Combining Overlay Networks and Cooperative Caching*

Frank T. Johnsen, Trude Hafsøe, Thomas Plagemann and Vera Goebel
Dept. of Informatics, University of Oslo, Norway
Research Report No 340, March 2006
ISSN 0806-3036
ISBN 82-7368-295-1
{frankjo, truhafso, plageman, goebel}@ifi.uio.no

ABSTRACT

The trends over the recent years show an increase in streaming media transported across the Internet in addition to already abundant web traffic, increasing the need for distribution schemes that efficiently handle this type of traffic. There are two obvious bottlenecks in any content distribution system, namely the origin server and the network. Our goal is to address both these bottlenecks simultaneously by using cooperative caching proxies on top of an overlay network. State of the art solutions within both subsystems utilize self-organization. Just combining these two can enable them to work together but may lead to a reconfiguration loop problem. In this report, we identify the challenges of combining these two subsystems, give an intuitive example illustrating the problem, and proceed to give an outline for several different solutions.

1. INTRODUCTION

In recent years there has been an increase in the types of data transmitted over the Internet from text and images to include multimedia data such as steaming of video and audio. Continuous data types need to be delivered to clients within a given time frame, thus placing high demands on the underlying transport system. Possibilities for interactivity, such as VCR-like operations on streams, further increase the demands on the distribution system. In addition to an efficient distribution scheme, the system must ensure responsiveness to client interaction. On-demand services, like News-on-Demand, combine regular web traffic with streaming media. Such services have demands on timeliness and synchronization, which are not originally supported in the Internet. Many solutions have been devised to address this problem, ranging from specialized router implementations and new network protocols to application level solutions. Most of these solutions aim to resolve one of the two potential bottlenecks in content distribution: The server and the network.

An overloaded server does not perform satisfactory and yields a poor user experience. Several solutions exist that attempt to remedy this, such as load balancing and reverse proxies. Furthermore, duplication of content on storage nodes located physically closer to the clients both offloads the origin server and reduces network traffic. The content duplication can be either the Akamai way, where the content owner pays for replication of specified content, or client driven caching in the case of caching proxies. The latter is especially important and in widespread use by ISPs, since caching popular content locally and thus limiting network traffic over other ISPs' networks lowers transmission costs.

The other potential bottleneck in content distribution is the network. Poor network performance can lead to long delays before the multimedia data arrives at the client, or make a service unusable by failing to deliver the data at all. The majority of network delays is caused by congestion in Internet routers, meaning one potential improvement in network performance is to implement support for multimedia streaming, priority routing and resource reservation in all Internet routers. This would be a very costly solution, and also require agreements between ISPs on how to perform this. These limitations prevent an Internet wide implementation of techniques that require router support, like for instance IntServ and DiffServ. Application level solutions do not suffer from this problem, while at the same time allowing for the deployment of enhanced distribution services, and have thus become the preferred way of improving network performance. Such solutions are often implemented as overlay networks because of the low cost and easy deployment associated with such networks.

Instead of incremental improvements that move the bottleneck from one subsystem to the other and vice versa, we aim to develop a distribution infrastructure which addresses both of these bottlenecks at the same time by using cooperative caching on top of an adaptive overlay network. Common for both these techniques is the recent trend towards dynamic, self-organizing systems which adapt to changing network conditions and changing workloads, i.e., how clients interact with the data. In the best tradition of systems design, like the layered system model, proxy caches and overlay networks are regarded in most of the existing solutions as separate and independent subsystems. This separation and the corresponding transparency implies that both subsystems can work together, but without any particular form of integration. However, a severe problem can arise when both dynamic subsystems do not take each others' dynamic nature into account. For example, if one subsystem changes

*This work has been performed in the context of the INSTANCE II project, which is funded by the Norwegian Research Council's IKT-2010 Program, Contract No. 147426/431.

.

its configuration due to observed changes in the network, a traffic shift will occur which may force the other subsystem to revise its view of the network. This change can lead to another traffic change, and may, in the worst case, make the system end up in a constant reconfiguration loop where the two subsystems spend a lot of time reconfiguring themselves. As current solutions focus on only one bottleneck at a time, none of them take this problem into account. The purpose of this report is to create awareness regarding this problem, and suggest a way to solve the problem by performing the enhancements in an integrated manner and not separately.

The remainder of the report is organized as follows: In Section 2, we present state of the art solutions of cooperative caching schemes and overlay networks. Section 3 presents the challenges that arise when integrating current modular systems with each other, and proceeds to suggest one possible way of resolving these challenges in Section 4. Section 5 concludes the report.

2. BACKGROUND

Proxy caching schemes and overlay network techniques have evolved since they were originally introduced. Early solutions were highly static in nature; both proxy caches and overlay networks were manually configured. Later, schemes evolved into being more dynamic in nature. Below we present a brief overview of the developments within these two areas of related work: Cooperative caching and overlay networks.

2.1 Cooperative caching

Caching is important to reduce network traffic, server load and improve end-to-end latency. The trend in proxy caching has been to evolve from stand alone caches to schemes where caches cooperate. The cooperation methods have been refined, and have shifted from static cooperating clusters to proxies forming dynamic cooperation groups reflecting on network changes and cost issues. This trend is evident both in classic web caching schemes and also in schemes for caching of streaming media.

2.1.1 Web caching

Originally, web proxy cache cooperation was done in a static manner by using techniques such as URL hashing, organizing the proxies into hierarchies, or by grouping proxies into pools with shared directories. See [6] for a detailed overview and discussions of past and current solutions. Many approaches presented there are not fully suitable for the scale of the Internet.

In the global Internet, origin servers may often be closer, better connected or more powerful than remote proxies, a fact that should be taken into account by the cache cooperation scheme. One such scheme is vicinity caching [5], which makes explicit choices between remote proxies and the origin server based on observed costs of fetching objects from various sources. In this approach each proxy defines a *vicinity* of other proxies relative to an origin server. This vicinity contains only those proxies it is preferable to contact and fetch objects from in addition to the origin server. If the server is distant and slow, the proxy will have a large vicinity, since many proxies would be preferable to the origin server. A fast, well connected site, on the other hand, may have a small or empty vicinity since fetching objects from the server would be faster than fetching from remote proxies. Each proxy tracks object locations only within the vicinity, and will not forward a request to another proxy in its vicinity unless its directory indicates that the object is cached there. The scheme measures latency of retrieved objects, and can thus estimate a latency based cost for retrieving objects from remote proxies and servers. As network conditions change the measured latency also changes, and vicinities grow and shrink accordingly. Such *dynamic vicinities* have the benefit of letting the cooperation scheme adapt to changes in the network characteristics as they occur. One can also modify the scheme to measure distance or other cost metrics and change the vicinities based on this, the importance is that each proxy and origin server is assigned a directly comparable numeric value.

2.1.2 Streaming media caching

Streaming media requires other caching solutions than those used for classic web content, an overview is given in [2].

The self-organizing cooperative caching architecture (SOCCER) [3] combines several orthogonal techniques. It employs a cost function which takes proxy load and distance into account to decide which proxies to interact with, making it a scheme suitable for use in a wide area network. SOCCER uses both caching and multicast techniques, and allows the proxies to form loosely coupled *cache meshes* for forwarding streaming data. Each proxy has a global view of the Internet, and chooses a subset of proxies to cooperate with. Since cooperation is based on a cost function, changes in proxy load or routes between proxies will affect the cache meshes. Cache meshes change dynamically to adapt to changing conditions. State information is distributed periodically as expanding rings, and the scope restrictions are based on varying the time-to-live field in the multicast packets. This ensures that the frequency of state messages exponentially decreases with increasing scope, while it still ensures global visibility over a larger time scale. Tests have shown that SOCCER yields the best overall performance when compared to hierarchical schemes, thereby indicating that self-organizing cooperation is a beneficial approach to streaming media caching on the Internet.

2.2 Overlay networks

Overlay networks were originally used as a means to increase scalability of distribution systems, in particular for streaming services, through the implementation of application level multicast. However, the low cost and ease of deployment of overlay networks lead them to be adapted for other purposes as well. Overlay networks developed from manual configuration to automatic systems for integrating new nodes into the network. The first challenge that was addressed was taking the placement of the nodes in the Internet into account when building an overlay network, so called topology awareness [7]. However, Internet network dynamics were not considered, leading to reduced overlay network performance when traffic shifts occurred in the Internet. This problem was resolved by introducing measurement-based overlay networks that automatically adapt themselves to changing network conditions. This adaptation can be done in one of two ways; either by changing which routes traffic follows over the already established virtual links, or by changing the virtual links themselves. Changing the setup of virtual links means modifying the *neighbor set* of each node. This set is maintained individually by each node, and consists of the set of nodes that this node has a direct virtual

link to. Each node uses this set to determine which nodes it needs to monitor and communicate with, and also which nodes to forward traffic to. The nodes in a neighbor set are selected based on their distance from the node maintaining that set. This distance is measured according to some network metric, for instance delay or bandwidth. Neighbor sets, and thereby also virtual links, will change over time to reflect changes in the measured network metrics. Common to most such self-organizing overlay networks is the use of a network monitoring service which can detect changes in the underlying network. Based on the results of the monitoring, the overlay network will adapt itself to the changing network conditions to improve the service to the user. Early overlay networks were designed to serve one specific purpose, and combined overlay networks with other techniques, forming very specialized solutions. In the past few years trends have turned towards making more general purpose overlay networks which do not support one specific application, but rather aim to function as a middleware layer forming a general purpose overlay. These generic overlay networks offer overlay routing and network optimization for a number of different network properties. One such service forms an overlay routing layer [4] which more specialized overlay networks can be built on top of. An even more generic solution is given in the Saxons [8] overlay network, which is a common overlay structure management layer designed to assist in the construction of large scale wide area Internet services. It dynamically maintains a high quality structure with low overlay latency, low hop count distance and high overlay bandwidth. However, overlay services which support both discrete and continuous media data may need a service which optimizes for several different metrics at the same time. The core idea in [1] is a self-organizing overlay network. This overlay network optimizes for several different QoS parameters at the same time by combining already established techniques like topology-awareness and network monitoring with an automated route maintenance mechanism that ensures efficient usage of Internet resources.

3. THE CHALLENGE

There are two important bottlenecks in a distribution system: The server and the network. We aim to build a distribution infrastructure which addresses both of these potential bottlenecks at the same time by applying cooperative caching on top of an adaptive overlay network. In the course of working towards this goal we have realized that a simple combination of these two dynamic techniques can have harmful effects, i.e., inefficient resource utilization and in the worst case unproductive reconfiguration loops. The resulting challenge is to coordinate these two self-managing subsystems in such a way that they can still be used independent of each other, but that the combination of the adaptations in both subsystems leads to the best resource utilization and system performance. In order to explain this challenge in more detail, we briefly describe the commonalities and differences between state of the art solutions in cooperative caching and adaptive overlays and use an example to explain how unproductive reconfiguration loops can occur:

- They rely on having an updated view of the network conditions on the Internet. Their knowledge of the underlying network comes from monitoring, often performed by an external monitoring service.

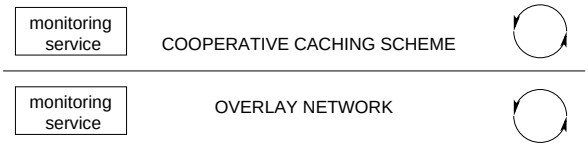

Figure 1: Modular approach with separate control loops and monitoring.

- Dynamic groups are utilized in order to adapt to changing network conditions while retaining scalability. In overlay networks a dynamic group is commonly referred to as a *neighbor set*. An overlay node's neighbor set consists of all nodes it has a direct virtual link to. A dynamic group in cooperative caching, on the other hand, is defined by the cost of communicating with other proxies. This means that there is no direct correlation between the neighbor sets and the *caching groups*.

- Both have a control loop that reconfigures the dynamic groups when needed. The need to reconfigure is determined by performing periodic evaluations of measured or observed network conditions. The control loop of the cooperative caching scheme does not necessarily evaluate the same network properties as the overlay network control loop does, but they still perform the same basic steps and make decisions based on either passive or active monitoring. Which type of monitoring is used varies from one solution to the next, but the end result is the same; an up to date overview of network conditions. Based on the results of monitoring a simplified and generic control loop will perform the steps shown below:

```
:control loop:
  1) check network conditions
  2) determine if conditions have changed
     enough to warrant reconfiguration
     - if yes, reconfigure
  3) wait
  4) repeat from 1)
```

The naive approach to concurrently address the server and network bottlenecks is to apply a cooperative caching scheme on top of an adaptive overlay network without any form of integration. Figure 1 illustrates this solution in which the two components operate independently, each with its own control loop and its own monitoring service. This leads to redundancy, because two monitoring systems are running at the same time and perform the same or similar tasks. The type of redundancy depends on whether the monitoring is active or passive. Active monitoring means that the monitoring service sends monitoring probes which it uses to calculate several different network properties, for instance bandwidth and latency. Passive monitoring can calculate the same properties, but does not send probes of its own. Instead it inspects passing traffic and performs calculations based on this. Active monitoring is often used in overlay networks, whereas passive monitoring is more commonly used by proxies.

3.1 Reconfiguration loops

The independence of the two control loops can lead to non-optimal resource utilization and to the severe problem of reconfiguration loops. Since the control loops are not aware of each other, they will make their decisions independently of each other. This may cause non-optimal solutions and even repeated reconfiguration which is triggered by the system entering a state as shown in the intuitive example presented below:

1. The system is in a steady state, illustrated in Figure 2. This figure shows seven overlay nodes connected by virtual links. We assume all these nodes are cooperating caching proxies forming dynamic cooperation groups. In this figure, we focus on one particular proxy, node 5, and its cooperation group (dotted oval). The load on each virtual link is denoted in three levels, *L*, *M*, and *H*, corresponding to low, medium and high load, respectively. The load levels are results of measurements, and thus encompass both overlay traffic and background traffic.

2. In Figure 3, there has been a reconfiguration of the overlay network. This has occurred because the overlay network control loop has detected a change in traffic on the Internet, for example because of a change in background cross traffic. This can be seen as a change in which virtual links are in use, in which case the medium load link between node 3 and node 4 is replaced by a link between node 3 and node 5. At the time of reconfiguration this link has a low load.

3. The reconfiguration of the virtual links leads to shifts in traffic. Traffic that previously was traveling across the old virtual link is now routed across the new one, leading to a medium load on the link between node 3 and node 5, see Figure 4. A good overlay network design should anticipate this change in traffic which is based on its own reconfiguration action, and does not attempt to reconfigure again at this point.

4. The caching scheme detects the reconfiguration done by the overlay network as a change in network conditions. This change means that the cost of communicating with node 1 and node 2 in Figure 4 is reduced, making them more attractive for node 5 to cooperate with. As a consequence, the proxy changes its dynamic groups as illustrated in Figure 5, causing a shift in the data that is transmitted over the virtual links. The new traffic pattern leads to an increase in load on several virtual links. The virtual links between nodes 1 and 3 and nodes 2 and 3 thus increase from a low to a medium load, whereas the virtual link between node 3 and node 5 now exhibits a high load. This increase in traffic can not be anticipated by the overlay network.

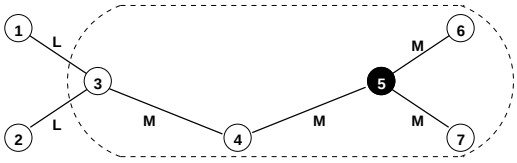

Figure 2: Initial steady state.

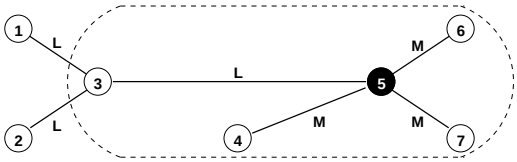

Figure 3: Virtual link change due to traffic shift.

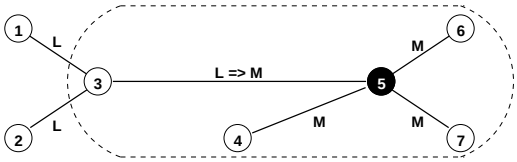

Figure 4: State after overlay network reconfiguration.

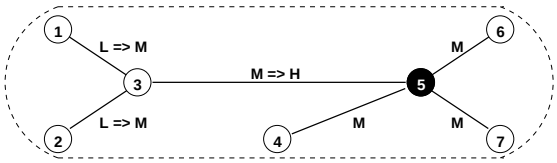

Figure 5: Caching scheme expands cooperation group.

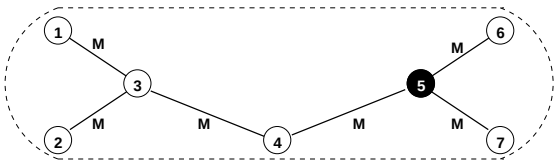

Figure 6: The overlay network reconfigures a virtual link.

5. The overlay network detects this increase in traffic as an unexpected change in network properties, and decides to reconfigure. Thus, it returns to the initial virtual link configuration, as shown in Figure 6. Because the cooperative cache dynamic group is unchanged at this point the load on the virtual links between nodes 1 and 3 and nodes 2 and 3 stays at a medium load level.

6. Node 5 now decides that the cost of utilizing node 1 and node 2 is too high, and excludes them from the

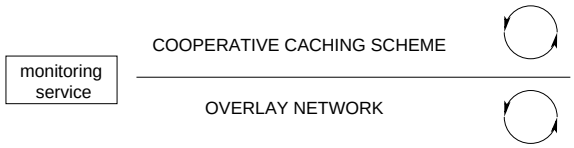

Figure 7: Modular approach with separate control loops and shared monitoring.

cooperation group. As a result of this, the traffic on the virtual links that connect node 1 and node 2 to node 3 will again decrease and return to their original low load. We are now back at the initial configuration as was shown in Figure 2. Due to this configuration of cache cooperation groups, the load measured by the overlay network is yet again favoring a reconfiguration. This leads us back to Figure 3, and we are in a reconfiguration loop.

This step-by-step example illustrates the reconfiguration loop problem that may arise when the overlay network and the cooperative caching scheme change their dynamic groups. In the example, the problem is shown for one proxy only, but the problem may be even bigger in practice as this may happen for multiple proxies at the same time.

4. SOLUTIONS

The two challenges identified in the previous section are those of redundancy and reconfiguration loop issues. The redundancy issue is minor and easily remedied, and we will address this issue first.

4.1 Monitoring redundancy

Redundancy is introduced into the system by having two monitoring services running at the same time. Different combinations of passive and active monitoring services give two different types of redundancy:

1. Calculation redundancy.
2. Probe redundancy.

By calculation redundancy we mean performing more calculations than are strictly necessary, for example by calculating the same bandwidth or latency between two nodes more than once. This wastes CPU cycles and uses more memory in each node than what is needed. Probe redundancy only occurs when both subsystems use active monitoring, since each monitoring service will transmit network probe packets, leading to an increase in network traffic.

The most common combination of monitoring services is likely to be that the cooperative caching scheme uses passive monitoring and that the overlay network uses active monitoring. A solution to reducing redundancy in this case would be to introduce a simple integration of the monitoring service, in which the active monitoring scheme is used by both overlay network and caching scheme, as illustrated in Figure 7. This form of integration may require the active monitoring service to be expanded to perform additional calculations to ensure that the needs of both subsystems are covered. The same is true for the opposite case where the overlay performs passive monitoring and the cooperative caching scheme uses active monitoring. If both subsystems use active monitoring, then one could replace both with a shared, extended active monitoring service which heeds the needs of both.

Overlay	Caching	Integration
Passive	Active	Keep active, share results.
Active	Passive	Keep active, share results.
Active	Active	Use one active, share results.
Passive	Passive	1) Keep both, share results. 2) Integrate into one passive. 3) Replace both with a shared active.

Table 1: Combinations of active and passive monitoring, and how to integrate them.

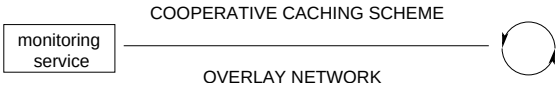

Figure 8: Modular approach with integrated control loops and shared monitoring.

There are several ways to integrate monitoring if both subsystems use passive monitoring. The three most obvious approaches are: If the two passive monitoring services base their calculations on different, non-compatible observed traffic, both must be kept. It is possible to optimize this by making sure that the same information is not calculated twice through the sharing of results. However, if the two passive monitoring services base their calculations on the same observed traffic, they can be integrated into one passive service which performs the calculations needed by both subsystems. The third option is to replace the two passive monitoring services with one common active service. This introduces some probing traffic into the system, but it allows for the calculation of more network properties than what the passive services can provide. An overview of combinations of the monitoring services that may be in use is given in Table 1 along with the suggested ways to perform integration.

4.2 Reconfiguration loop avoidance

Integrating the monitoring solves the redundancy issue, but the reconfiguration loop problem still remains. We have identified five disjunct approaches concerning the problem:

1. Ignore the problem.
2. Integrate the control loops.
3. Make the control loops mutually aware of each other.
4. The cooperative caching control loop is aware of the overlay network.
5. The overlay network control loop is aware of the cooperative caching scheme.

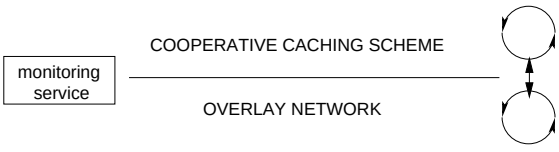

Figure 9: Modular approach with mutually aware control loops and shared monitoring.

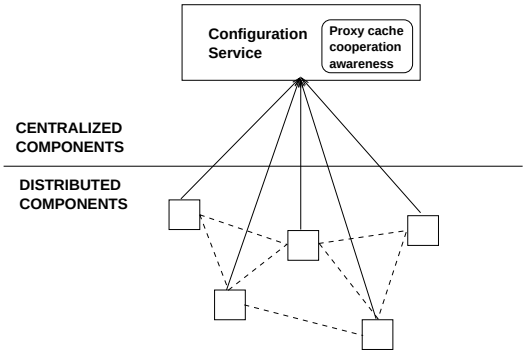

Figure 10: Overlay network with a centralized configuration service capable of taking proxy cache cooperation scheme decisions into account.

The first, simplest and most naive approach is using the ostrich algorithm and just ignore the problem. This method may be sufficient if the problem described in Section 3 rarely occurs. Using this approach requires caution as it is difficult to be sure that the problem will not lead to indefinite reconfiguration loops, as this may render the system unusable. In addition, ignoring the problem gives a less than optimal resource usage, and we opt for other solutions.

Figure 8 illustrates the second approach of fully integrating the control loops. This approach requires heavy modification of central parts of each scheme, leading to a shared control loop which handles both cooperative caching and overlay network reconfiguration needs. In essence, this means creating a new specialized solution where overlay and caching is tightly coupled. This tight coupling limits the usability and future development possibilities of the system, but redundancy is eliminated and the reconfiguration loop issue is resolved.

The third approach, shown in Figure 9, loosely couples two separate subsystems by letting their respective control loops be mutually aware of each other. In this case the schemes retain much of their modularity, which allows for easy expansion and modification at a later time. Because there are two control loops there is some redundancy in communication overhead and state information, but the reconfiguration issue can be solved. However, there is an increase in complexity, since each control loop needs to process and take into account the information received from the other. Also, the issue of agreement arises: What if the control loops do not agree on the best configuration of the network? This requires well defined interaction rules, in which one control loop is defined to be the tie breaking one in such cases.

Approaches 2 and 3 represent two extremes. Between these two there are other possibilities, namely approaches 4 and 5, where only one reconfiguration loop is aware of the other. Using one of these approaches avoids the downsides of tight coupling by allowing the system to remain modular, while at the same time limiting complexity and overhead. Thus, both these two are good candidates for reconfiguration loop avoidance.

In approach 4, the cooperative caching scheme is made aware of the overlay network. This awareness comes in one of two ways:

1. Direct communication between the two subsystems.
2. Anticipation of reconfiguration based on knowledge of the other subsystem's decision logic.

Direct communication means that the overlay network control loop informs the cooperative caching scheme about all the reconfiguration decisions it makes, thus allowing the cooperative caching scheme to take this information into account when performing its own reconfiguration. The second option means that the overlay network does not share all the decisions made, it simply informs the caching scheme once about the logic it uses to make these decisions. Assuming that the caching scheme has full knowledge of this logic, and in addition the same knowledge about the underlying network as the overlay network, it can predict which actions the overlay network will perform. This means that the task of reconfiguration loop avoidance in this case falls on the cooperation scheme. However, exporting logic in this direction may not be a good solution, because the overlay network will in most cases have a more complete view of the network than the cooperative caching scheme. This means that a large amount of data must be relayed from the overlay network to the cooperative caching scheme control loop to enable it to correctly anticipate which actions will be performed. Just informing the cooperation scheme about the reconfiguration performed may not be sufficient to avoid the reconfiguration loop problem. This is due to the fact that most overlay network solutions keep a limited view of the network in each node, i.e., it knows about its own virtual links only. Thus, local decisions passed up to the cooperative scheme control loop do not really help in solving the problem. This control loop needs a broader overview of the participating nodes in order to detect and avoid the problem. As a consequence of this, the overlay network one employs in a cooperative caching content network needs to be aware of a wide area of the network, preferably having global knowledge.

Approach 5 is the inverse of the previous approach, here the overlay network control loop is aware of cooperative caching group dynamics. As above, there are two ways to obtain awareness, namely direct communication or exporting logic. This approach is better than the previous one, because it avoids many of the difficulties mentioned above. If the overlay network has a broader view of the underlying network, a simple solution to the problem springs to mind: By exporting the cooperative cache reconfiguration logic to the overlay network, the overlay network control loop can utilize its knowledge of the data to reliably anticipate and thus avoid reconfiguration loops.

The awareness enabled control loop should preferably have a global view of the network in order to best utilize this information to determine and compensate for possible shifts in traffic. We focus on approach 5, and outline a loop avoidance solution based on an overlay network solution that lends itself well to this type of modification. The overlay network solution that we want to expand to encompass loop awareness combines distribution and centralization principles [1]; critical tasks like monitoring, routing and data transport are performed individually on each node, while the reconfiguration mechanism is centralized. This centralized configuration service has a global view of the network which it uses to make all reconfiguration decisions. If this configuration service can be made aware of the principles for dynamics in cache cooperation groups it can anticipate the following traffic shift effect, thus allowing it to foresee and avoid reconfiguration loops. Such a change to the configu-

ration service is relatively minor compared to approaches 1 through 4, as all nodes in the network remain unchanged. Figure 10 illustrates this solution.

5. CONCLUSIONS

In this report, we have looked into combining two established techniques for bottleneck avoidance in distribution systems. We have identified challenges which arise when combining two autonomous, self-organizing techniques; cooperative caching and an overlay network. The main challenge is to overcome a possible reconfiguration loop which may occur when both subsystems are self-organizing. We have identified five approaches that address this problem, and have discussed their suitability in that respect. One solution we found to be particularly interesting is that of an overlay network which is aware of cache cooperation scheme logic.

In addition to resolving the reconfiguration loop issue, one also needs to look into various compromises for avoiding the monitoring service redundancy. If one is to replace the proxy cache and overlay network monitoring service with a shared resource, it is essential that a compromise that works for both subsystems is found.

6. ACKNOWLEDGMENTS

This work was initiated while F. T. Johnsen and T. Hafsøe were visiting faculty at the University of Twente in Enschede, the Netherlands. We would like to thank Marten van Sinderen and the ASNA group for their hospitality.

Furthermore we would like to thank Katrine S. Skjelsvik at the University of Oslo for proofreading the report and providing us with valuable feedback during the writing process.

7. REFERENCES

[1] T. Hafsøe, T. Plagemann, and V. Goebel. Towards Automatic Route Maintenance in QoS-Aware Overlay Networks. Technical Report 329, University of Oslo, November 2005. ISBN 82-7368-284-6.

[2] M. Hofmann and L. Beaumont. *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann, March 2005. ISBN 1-55860-834-6.

[3] M. Hofmann, E. Ng, K. Guo, S. Paul, and H. Zhang. Caching techniques for streaming multimedia over the internet. Technical report, Bell Laboratories, April 1999. BL011345-990409-04TM.

[4] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. In *Proceedings of the ACM SIGCOMM Conference*, August 2003.

[5] M. Rabinovich, J. Chase, and S. Gadde. Not all hits are created equal: cooperative proxy caching over a wide-area network. *Computer Networks and ISDN Systems*, 30(22–23):2253–2259, 1998.

[6] M. Rabinovich and O. Spatscheck. *Web caching and replication*. Addison Wesley, 2002. ISBN 0-201-61570-3.

[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM'02*, 6 2002.

[8] K. Shen. Saxons: Structure management for scalable overlay service construction. In *USENIX NSDI '04*, pages 281–294, 2004.