Minicamp: Prototype for Partial Participation in Structured Peer-to-Peer Monitoring Protocols

Andreas Disterhöft
Technology of Social Networks, University of Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf, Germany
Email: disterhoeft@cs.uni-duesseldorf.de

Kalman Graffi
Technology of Social Networks, University of Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf, Germany
Email: graffi@hhu.de

***Abstract*—Quality of service and quality of experience are of increasingly interest for successful communication network applications, particularly for decentralized networks, such as peer-to-peer (p2p) networks. They need sophisticated monitoring mechanisms to be able to adapt the system's parameters to maintain a certain level of quality of service. In the literature, several well-studied approaches, classified as structured monitoring overlays, have been proposed. To the best known of the authors, these do not tackle the problem of partial participation. With Minicamp we propose a middleware which operates between the p2p overlay and the monitoring overlay and creates the opportunity to run monitoring services on just a subset of all nodes. These nodes gather and disseminate monitoring data of the whole system which is achieved by probing the subset of participating nodes. Evaluations reveal high precision for the probing scheme even when facing low participation rates.**

I. Introduction

Quality of service is an important basis for the users' experience which participate in applications operating on top of certain communication networks. Being able to hold a certain level of quality is crucial for such networks, especially, if the network is decentralized, that is, operating in a peer-to-peer (p2p) fashion. Proposed state-of-the-art structured monitoring approaches, such as CONE [3], SDIMS [13], SOMO [14] and SkyEye.KOM [8], [9], tackle this problem and provide information about the current state of the network. This information can be used to dynamically adapt the overlay's parameters accordingly, as suggested by Graffi [7]. Besides being very precise, these approaches assume, to the best knowledge of the authors, that all nodes participating in the network also participate in the monitoring approach. This is not necessarily true as nodes may operate defective, that is, because the desired feature is not yet implemented in the user's software version or the software behaves maliciously. In such cases, the building and maintaining process of the monitoring overlay's highly desired structure, for example a tree, may be hindered or completely disturbed, which ultimately leads to a severe loss of data and monitoring precision.

In this paper we tackle the problem of partial participation in the monitoring and propose a first prototype, which obtains reliable and precise monitoring data of all participants, even in scenarios with low participation. We define passive nodes as peers, which are not aware of the monitoring feature, and active nodes as peers, which support the monitoring feature. The existence of passive nodes in communication systems is motivated by the problem of software updates and its badly desired installation. An example is Java [2], where only 50% of the users updated to the most recent version, which is Java 8 and has been released three years ago. Another famous example is the software version share of Windows [1]. Just like these examples, a p2p software might introduce a monitoring feature with its latest release and peers, who updated their version, become active nodes.

We propose Minicamp, a *m*iddleware for *in*complete part*ic*ip*a*tion in *m*onitoring *p*rocotols, with following aims:

A1) Provide a functionality to support the monitoring of active nodes within the p2p overlay
A2) Monitor both active and passive nodes in order to provide a full global view on all active nodes and a limited global view on passive nodes
A3) Offer the possibility to reuse current monitoring solutions

In a nutshell, our proposed middleware Minicamp is an overlay structure running between a common p2p overlay and an existing state-of-the-art structured monitoring solution. Thus, Minicamp organizes active nodes, obtains additional monitoring information from passive nodes and distribute a common global view via an ordinary monitoring approach.

The remainder of the paper is structured as follows. We present our three-point solution Minicamp in Section II. First, a mapping overlay is created, which allocates passive nodes to active nodes. Then, a probing mechanism is proposed to continuously probe passive nodes. Lastly, this obtained data base is used to inject the data, together with the active node's information, into a common structured monitoring approach. By doing so, all active nodes aggregate and disseminate monitoring information of the whole p2p overlay. After the description of Minicamp, we evaluate its performance in Section III. We conclude our work in Section IV.

II. Our Solution: Minicamp

Our proposed protocol Minicamp is described in the following. In environments of incomplete participation it allows nodes that are aware of the monitoring mechanism (active nodes) to gather and disseminate monitoring information. For this, active nodes scan for passive nodes in a middleware overlay and atop of this overlay an existing structured monitoring

This work was supported by the DFG Grant GR4498 1-1 - OverlayMeter

overlay is used to maintain a global view for both active and passive nodes. Our protocol states the following requirements:

R1) The p2p overlay should implement the KBR-Interface [4].
R2) The p2p overlay uses an one-dimensional ID space or can be translated to such, e.g. use space-filling curves [11].
R3) Node and object IDs must be located within one ID space
R4) Convex node responsibility range: if node A with ID id_A answers on a lookup of the ID id_{lookup} then the node A is at least responsible for all IDs between id_A and id_{lookup}

Minicamp is divided into three mechanisms, which are described as steps in the following, each runs in parallel and uses the interfaces of its predecessor.

A. Step 1 - Creating Middleware Overlay: Join and Scan

In the first step, active nodes create and join an independent p2p overlay, for example, Chord [12]. In this middleware overlay active nodes $\mathbb{S}_{active}$ subdivide the whole ID space into disjoint ranges for which they are responsible to effectively scan passive nodes $\mathbb{S}_{passive}$. Their task is to keep the scanning range current. From a top view, active nodes in its entirety try to obtain the set Π_α specified by (1). In the scope of this paper we assume that such a mechanism is given, more precisely, (1) is approximated by such.

$$\forall\, \alpha \in \mathbb{S}_{active}\,,\, \Pi_\alpha = \left\{ \rho \mid \quad \rho \in \mathbb{S}_{passive} \wedge id_\rho \in \phi_{scanRange}(\alpha) \right\} \quad (1)$$

B. Step 2 - Probing: Obtain Data from Passive Nodes

In the second step, the actual probing of the passive nodes is taking place. For this, we assume a middleware overlay has been built up and the scan set (1) is obtained with a certain degree of accuracy. While the middleware keeps the set of passive nodes current, the probing mechanism passively obtains monitoring information from them. Therefore, active nodes send probing messages, for example pings, and interpret their response. The interpretation of the response implies an extraction of monitoring data. This procedure is necessary as passive nodes are not willing or able to answer direct requests as they do not support the deployed monitoring mechanism. We provide interfaces to allow adding additional probing mechanisms.

Probed monitoring data is stored locally in a transient storage space, which is made available through an interface for further monitoring purposes. In summary, see (2), each active node tries to probe the local monitoring view δ_{γ_t} of passive nodes γ it is responsible for at time t. The probing is optimal, if ϵ is minimal, thus, the probed view is most current.

$$\forall\, \alpha \in \mathbb{S}_{active}\,,\, \Delta_{\alpha_t} = \left\{ \delta_{\gamma_{t+\epsilon}} \mid \gamma \in \Pi_\alpha \wedge \epsilon > 0 \right\} \quad (2)$$

C. Step 3 - Monitoring: Apply Common Monitoring Overlays

The last step is the monitoring which builds its structured overlay on top of the middleware overlay. It is supposed to run conventional monitoring approaches like SkyEye.KOM [9]. Thus, only active nodes participate in such monitoring mechanisms and inject their own view and the views of their probed passive nodes by using the aforementioned transient storage interface of step 2 (Sec. II-B). Utilizing the components to aggregate and disseminate monitoring data, we are able to create a global view including both active and passive nodes.

III. Evaluation

In the scope of this paper, we demonstrate a proof of work and show an excerpt of the evaluation of Minicamp considering the following metrics:

- Precision of the monitoring approaches' global view, which includes realistic probed information of passive nodes, compared to a perfect probing mechanism (100% precision) and compared to the reference where all nodes are active, thus, no probing mechanism is required.
- Traffic costs of the probing mechanism compared to the regular traffic of the monitoring overlay.
- Freshness of the monitoring approaches' global view compared to the reference with no passive nodes (no probing mechanism is required).

A. Methodology

To evaluate the aforementioned metrics, we implemented a prototype of Minicamp in PeerfactSim.KOM [5]. This implementation contains a mapping approach which approximates Step (1), but its performance is not subject of this paper due to space limitations. We refer to a fixed active node share of 1%, which is meant to be a worst case scenario. This share is motivated by the distribution of software version shares, whereas, a 1% active node share represents a fresh release of a software version. We ran the structured monitoring overlay SkyEye.KOM [9] atop a probing mechanism which is described as follows.

In a nutshell, we made use of a probing mechanism to probe the bandwidth of passive nodes by taking the Internet delay into account and manipulate the exact value accordingly. This is motivated by real world bandwidth probing approaches such as from Goebel et al. [6]. The authors propose to send packet trains through a network to a receiver and interpret the answer by consulting the delay and payload. In combination with a Weibull distribution characterizing the Internet delay, we are able to simulate the probing error for our simulations. Such distribution is proposed by Hooghiemstra et al. [10], which is modeled for investigated routes across the Internet.

For the monitoring overlay SkyEye.KOM, participants gather information from two data provider classes. The deterministic data provider is used by the participants to model a function which solely depends on time, e.g. sinus, and thus is ideal as reference. Whereas, with the real data provider, nodes raise real metrics, e.g. the lookup hop count of the underlying p2p overlay. To scale the Weibull distributed value onto the data provider's domain, we propose to drop a percentage of the heavy-tailed distribution and apply drop policies. As most real data providers live in the domain of $[0;\infty]$, or $[-\infty;\infty]$, no scaling is applied. For bounded domains, as the sinus function ($[-1;1]$), we evaluate different drop strategies. The

TABLE I
EVALUATION OF MONITORING PERFORMANCE - WORST CASE WITH 1% ACTIVE NODE SHARE.

Scenario		Metric		Metric		Metric		Metric		Metric	
Monitor function, Drop Percentage, Drop Policy ([N]o, [R]eroll)	[D]efault [F]ull [P]artly	Count (Abs. Error)		Count (Rel. Error)		Max (Abs. Error)		Min (Abs. Error)		Mean (Abs. Error)	
		Ø	σ	Ø	σ	Ø	σ	Ø	σ	Ø	σ
Sinus, 0, N	D	132.5	350.2	-0.018	0.092	0.740	0.726	0.738	0.720	0.571	0.298
	F	496.2	315.3	-0.116	0.079	0.522	0.552	0.507	0.533	0.260	0.142
	P	496.2	315.3	-0.116	0.079	16.236	5.192	0.873	0.563	0.260	0.143
Sinus, 0, R	D	113.4	220.2	-0.012	0.061	0.748	0.731	0.754	0.730	0.565	0.286
	F	362.8	274.8	-0.083	0.062	0.513	0.547	0.506	0.537	0.254	0.145
	P	362.8	274.8	-0.083	0.062	0.996	0.707	0.696	0.578	0.346	0.194
Sinus, 0.01, R	D	127.1	325.8	-0.018	0.082	0.747	0.727	0.749	0.727	0.559	0.282
	F	404.2	268.5	-0.094	0.060	0.500	0.534	0.504	0.533	0.245	0.138
	P	404.2	268.5	-0.094	0.060	0.993	0.702	0.569	0.563	0.271	0.118
LookupHopCount, 0, N	D	119.1	245.9	-0.015	0.066	3.253	0.140	1.362	0.163	0.011	0.008
	F	469.2	347.6	-0.110	0.081	3.272	0.129	1.379	0.174	0.010	0.010
	P	469.2	347.6	-0.110	0.081	16.021	5.325	1.778	0.171	0.017	0.014

Reroll strategy forces a reroll if the value is not in the bounded domain.

We compare the *Default* scenario, which is our reference, with the *Full* and *Partly* scenarios. *Default* contains 100% active nodes, thus, our approach Minicamp is not applied. *Full* applies Minicamp with a 100% precision probing mechanism, where probed passive nodes always answer with their exact value. *Partly* applies Minicamp and uses the probing mechanism described before. All scenarios are simulated ten times with 5000 nodes performing atop Chord. The evaluation results are shown in Tables I–II and are discussed below.

B. Monitoring: Precision

For the evaluation of the monitoring performance, especially its precision and freshness of data, we refer to Table I. Scenarios are presented row by row and relevant metrics are presented in columns. Per metric the average value and its standard deviation throughout the monitoring phase of the simulation is shown.

The first two columns present the absolute and relative error of the count attribute, respectively, which is the number of involved data sets in the aggregated global view from both, active and passive nodes. The *Full* and *Partly* scenarios encounter around 3 to 4 times more loss than the reference *Default* scenario. In relative numbers to the desired count this equals 5 to 7 times higher relative loss. This mainly depends on the used mapping approach, which we have chosen initially, and is independent of the probing approach, thus, *Full* and *Partly* scenarios perform equally.

The third column is the maximum metric, which is the absolute error to the desired maximum value that has been aggregated up the monitoring tree, thus, it is an indicator of data's freshness. *Partly* scenario without any drop policy faces a lot of overestimations, 16.2 for sinus and 16 for lookup hop count in average, and a very high standard deviation (cf. around 5). For the reroll policy this effect shrinks but *Partly* still performs worst. *Full* performs better than *Default* and *Partly* slightly worse than *Default* (cf. around 0.7, 0.5 and 1.0). The reason why *Full* performs better than *Default* is because of the mapping overlay's feature of having a smaller tree depth than the reference tree as only active nodes are participating in it. Therefore, in the *Full* and *Default* scenario 50 and 5000 nodes, respectively, participate in the monitoring tree. The numbers are meant to be the maximum participating nodes as they are subject to churn.

The fourth column is the minimum metric, which represents the absolute error of the minimum value in the aggregated view and its meaning is analogous to the maximum metric. Overestimations are encountered for real and deterministic data using no drop policy, which are not in the magnitude of the maximum metric's values. For real data, we face an increase of the average error from the reference's value of 1.36 to a value of 1.38 (*Full*) and 1.78 (*Partly*), whereas the standard deviation is not significantly affected. Interestingly, just like in the maximum metric, the *Full* scenario does not perform better than the *Default* scenario, which is most likely due to missing data sets (cf. count metric). Applying the reroll drop policy on the deterministic function, the error of *Full* and *Partly* scenarios undercut the error of the *Default* scenario, while the error of the *Partly* scenario decreases further when additionally applying a 1% tail drop.

The last column is the absolute error of the aggregated data's mean, which is the most important metric in terms of precision. We observe almost equal absolute errors for all three scenarios with the *Partly* scenario being a bit more shaky (higher standard deviation), which is due to the aforementioned overestimations of max and min values. For deterministic data applying no drop policy, we encounter that *Full* and *Partly* scenarios perform almost exactly the same, while the reference performing worse. In *Default*, more nodes participate in monitoring, thus the monitoring tree is deeper leading to a higher monitoring delay. Applying the reroll policy with no tail drop, we observe that *Partly* suffers under higher errors and standard deviation but still operates below the *Default* scenario. When applying tail drop, this effect shrinks.

We conclude that the monitoring performance suffers under the provided mapping overlay and the probing of data but is able to cope with the reference scenario in terms of precision.

TABLE II
EVALUATION COSTS FOR 1% ACTIVE NODE SHARE.

Message type	Throughput per peer throughout the simulation [$\frac{byte}{s}$]			
	$\emptyset$	$\sigma_\emptyset$	share > 0 [%]	max
all	16791.40	8438.50	1.00	79806.78
monitoring	39.06	27.79	1.00	272.09
probe req.	3.33	47.04	0.01	1504.66
probe resp.	3.02	1.23	0.98	4.29

C. Costs: Bandwidth

Table II shows the outgoing bandwidth throughput per peer throughout the whole simulation running all three scenarios in parallel for the worst case of 1% active node share sorted by certain message types. These message types are probe requests, which are sent by active nodes running the probing mechanism, probe responses, which are sent by passive nodes, monitoring messages, which are part of the monitoring overlay SkyEye.KOM and all arising messages aggregated, which, besides the previous three message types, include overlay maintenance and lookup messages. The table shows the average bandwidth and its standard deviation, the share of nodes experiencing traffic and the maximum traffic a single peer faces. This data take the mapping overlay's traffic into account but it is not discussed. Noticeable is the unfair distribution of sent probe requests and responses (cf. 1% and 98%). Though, the node with the highest load faces the 451 fold of the average traffic, it is still low in absolute numbers ($1.5 \frac{KByte}{s}$). The high standard deviation (50% of the average traffic) indicates a wide spread around the average. Thus, the node facing the most traffic is, relative to the wide spread around the average, still low, facing the 4.8 fold of the average traffic. We conclude, that active nodes with a low bandwidth might get overloaded in worst case scenarios where the active node share is low.

D. Costs: Delay

Besides the investigation of traffic costs, we evaluate delay costs of the monitoring overlay in this section. We distinguish between the aggregated data delay, which is the time needed to traverse the monitoring tree bottom-up aggregating all relevant data, and the delay of the global view, which is the final aggregation of all data disseminated top-bottom. The influence of Minicamp on these delays can be reduced to its influence on the monitoring tree's height. Data taken from simulations of a 1% active node share scenario (= *Full / Partly*) compared to a 100% active node share scenario (= *Default*) in an environment without churn reveal the following numbers. For the aggregated view we faced a mean / max duration of 82.8 / 228.1 vs. 36.0 / 112.5 seconds and for the global view we measured a mean / max duration of 108.8 / 272.0 vs. 64 / 131.0 seconds, where the first tuple is meant to be the *Default* and the second the *Full / Partly* scenario. We wrap up: The smaller the active node share, the smaller the monitoring tree and the fresher the monitoring data. The former also implies a higher active node load, as mentioned in Section III-C.

IV. CONCLUSION AND FUTURE WORK

In this paper we propose the prototype Minicamp, a three-stage solution to tackle the problem of partial participation in structured monitoring approaches. The three stages contain the allocation of so called passive nodes to active nodes in the mapping overlay, the probing of the passive nodes by active nodes and the aggregation and dissemination of complete monitoring data in an ordinary structured monitoring system like SkyEye.KOM [9]. The complete monitoring data contain relevant data from active nodes and probed information from passive nodes. First worst case evaluations of the monitoring performance show that Minicamp is capable to provide high precision data. The costs are manageable and depend on the actual active node share. For future work, we provide an in-depth analysis of the Minicamp mapping overlay and evaluate its performance and capabilities.

ACKNOWLEDGMENT

We would like to thank Franz Kary for his great work. Computational support and infrastructure was provided by the Centre for Information and Media Technology (ZIM) at the University of Düsseldorf (Germany).

REFERENCES

[1] "NetMarketShare: Operating Systems' Share," https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0%20, accessed: 2017-04-16.

[2] "Plumbr Java Version Share," https://plumbr.eu/blog/java/java-version-and-vendor-data-analyzed-2016-edition, accessed: 2017-04-16.

[3] R. Bhagwan, G. Varghese, and G. Voelker, "CONE: Augmenting DHTs to Support Distributed Resource Discovery," Tech. Report CS2003-0755, University of California, San Diego, 2003.

[4] F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *IPTPS '03: Proc. of the Int. Workshop on Peer-To-Peer Systems*, ser. Lecture Notes in Computer Science, vol. 2735. Springer, 2003.

[5] M. Feldotto and K. Graffi, "Comparative Evaluation of Peer-to-Peer Systems Using PeerfactSim.KOM," in *IEEE HPCS'13: Proc. of the Int. Conf. on High Performance Computing and Simulation*, 2013.

[6] N. Goebel, T. Krauthoff, K. Graffi, and M. Mauve, "Moving measurements: Measuring network characteristics of mobile cellular networks on the move," *Computer Communications*, vol. 97, pp. 81–95, 2017.

[7] K. Graffi, "Monitoring and management of peer-to-peer systems," Ph.D. dissertation, Technische Universität Darmstadt, 2010. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/2248/

[8] K. Graffi and A. Disterhöft, "SkyEye: A tree-based peer-to-peer monitoring approach," *Pervasive and Mobile Computing*, 2017.

[9] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, "SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems," in *Proc. of the Int. Conf. on Parallel and Distributed Systems*. IEEE, 2008.

[10] G. Hooghiemstra and P. V. Mieghem, "Delay Distributions on Fixed Internet Paths," Delft University of Technology", Technical Report 20 011 020, 2001.

[11] G. M. Morton, *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York, 1966.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proc. of ACM SIGCOMM '01*, 2001, pp. 149–160.

[13] P. Yalagandula and M. Dahlin, "Research Challenges for a Scalable Distributed Information Management System," The Univ. of Texas at Austin, Dept. of Computer Sciences, Tech. Rep. CS-TR-04-48, 2004.

[14] Z. Zhang, S. Shi, and J. Zhu, "SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT," in *IPTPS '03: Proc. of the Int. Workshop on Peer-To-Peer Systems*, 2003.