Dishonest Reporting in Queue-Based Cross-Layer Network Optimization

Dimitris Giatsios Iordanis Koutsopoulos Thanasis Korakis

University of Thessaly and CERTH, Greece

***Abstract*—Queue-based cross-layer optimization algorithms have recently been a subject of intensive research in wireless networks. Their purpose is to guarantee stable operation and to achieve some form of fairness among users, whenever the traffic demand exceeds network capacity. Despite the plethora of work in this field, the scenario where one or more nodes declare false queue backlog values in order to gain throughput advantage remains unexplored. In this paper we examine this type of selfish misbehavior, concentrating on a specific class of algorithms, the so-called quadratic Lyapunov-function-based algorithms (QLA). In particular, the effect of backlog misreporting on a single-hop access network with contending stations is evaluated through simulations. A simple framework for the detection of misbehaving nodes is proposed, under the assumption that the access-point is aware of the utility functions of the stations. The detection approach exploits the fact that under QLA the throughput of a node must be approximately equal to an "expected" value, derived from the reported queue backlogs.**

I. Introduction

Cross-layer optimization algorithms for wireless networks have received much attention during the last decade. Use of queue backlog or head-of-line delay information has been proven to give throughput-optimal resource allocation algorithms. To provide some form of fairness when the total traffic demand exceeds network capacity, utility-based flow control algorithms have been developed, which run on the nodes and interact with the scheduling and resource allocation component, in order to jointly maximize the aggregate utility. A class of such cross-layer algorithms are the Quadratic Lyapunov-function-based Algorithms (QLA), developed in [1], [2].

Despite the plethora of work in this field, a specific aspect has been overlooked. It is assumed that all nodes truthfully report their backlogs to the scheduling component. However, a node has a selfish motive to misreport. See, for example, Figure 1, where two wireless stations are transmitting traffic to an access-point. The link capacities are time-varying and at each timeslot the scheduler decides which station will transmit based on the max-weight algorithm (see [3], [4]), that selects the node with the greatest product of link capacity and queue backlog. If the queue backlogs and the channel capacities have the values depicted, the access-point will schedule station B to transmit. If station A lies about its backlog declaring 20 packets instead of 10, it will be the one scheduled instead.

978-1-4673-1298-1/12/$31.00 © 2012 IEEE

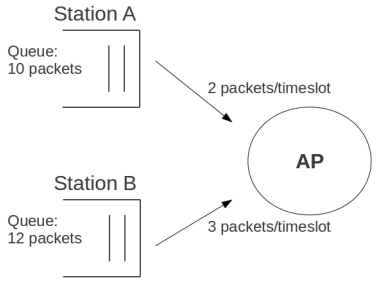

Fig. 1. Simple example of access network susceptible to misbehavior

In cross-layer architectures the typical approach for tackling such behavior is through dynamic pricing techniques. A typical example is the VCG mechanism (see [5], [6]), through which an optimal strategy for a rational user is to reveal its true utility function, but the algorithm complexity is prohibitive for implementation. Schemes with prices determined by user bids have been proposed in [7], [8]. In [9] dishonest reporting of link capacity is examined and a heuristic with a linear pricing function is proposed as countermeasure.

While pricing mechanisms are the usual approach for clean-slate cross-layer optimization algorithms, detection-oriented approaches have been proposed as amendments to existing or readily implementable algorithms. See [10], for example, where uncooperating users in multi-hop ad-hoc networks are detected, or [11], where 802.11 MAC misbehavior is detected.

In this paper, we examine a specific type of selfish behavior not explicitly studied before, to the best of our knowledge. This is false reporting of queue backlogs in cross-layer optimization with QLA. We focus on the case of cellular access networks and evaluate the effect of misreporting on network performance. We propose detection algorithms in the cases of infinite traffic demand and arbitrary input rates, under the assumption of knowledge of the utility functions.

II. Network model

We consider a network with N wireless stations and an access-point, as in figure 2. We examine uplink traffic only, as the type of misbehavior we study does not affect the downlink. Each of the N links has a time-varying capacity $C_i(t)$. Arrivals $A_i(t)$ at the transport layer of each node are Poisson processes with averages $\lambda_i(t)$. A flow controller at each node determines the amount $R_i(t)$ to be admitted to the

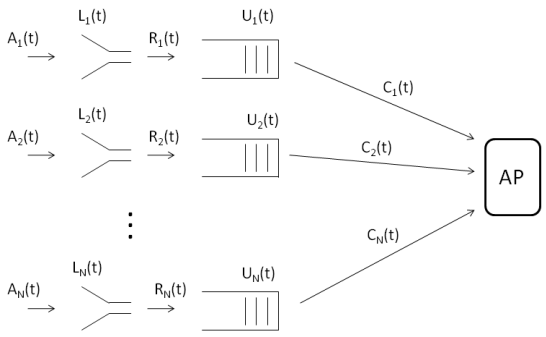

Fig. 2. Network model

network layer during each timeslot. Unadmitted data is kept in transport layer reservoirs with backlogs $L_i(t)$. Admitted data enters network layer queues with backlogs $U_i(t)$.

Only one station can transmit during each timeslot. The access-point decides the station $I(t)$ to transmit based on the max-weight principle [3]:

$$I(t) = \arg\max_{i=1,\ldots,N} U_i(t)C_i(t) \tag{1}$$

Whenever the vector of average arrival rates is within the network capacity region, the above algorithm guarantees queue stability [4]. To ensure stable operation when arrival rates exceed the capacity region, the flow controllers must ensure the admitted data rates are kept within the boundaries of the capacity region. Furthermore, fair throughput allocation among nodes must be guaranteed, where fairness is usually expressed as maximization of the sum of utilities of the nodes. We assume the utility functions $g_i(\cdot)$ are smooth and concave.

In general, the network planner must solve the optimization

$$\text{Maximize:} \qquad \sum_{i=1}^{N} g_i(r_i) \tag{2}$$

$$\text{subject to:} \qquad (r_i) \in \Lambda, \tag{3}$$

$$0 \leq (r_i) \leq (\lambda_i), \tag{4}$$

where r_i is the long-term throughput of node i and Λ is the capacity region, not known a priori for a wireless network.

Two algorithms satisfying these criteria have been proposed in [1] for the cases of infinite and finite traffic demand respectively. With the first term we describe the assumption that all the nodes have always enough data in the transport layer to be admitted to the network layer, so the constraint (4) can be omitted. Dropping this assumption and allowing for arbitrary input rates, (4) becomes active and the techniques for solving the optimization problem are more complicated.

For infinite demand, the flow control algorithm is called FLOW1 and works as follows. Each timeslot, the flow controllers choose $R_i(t) = r$, where r solves the maximization:

$$\text{Maximize:} \qquad Vg(r) - rU_i(t) \tag{5}$$

$$\text{subject to:} \qquad r \leq R_i^{max}, \tag{6}$$

where R_i^{max} is a burstiness constraint for node i and V is a parameter determining the tradeoff between proximity to the maximum utility sum and aggregate network congestion.

For finite demand, auxiliary variables $\gamma_i(t)$ and virtual queues $Y_i(t)$ are used in order to solve the optimization with the extra demand constraint. The relative algorithm, called FLOW2, works as follows. The amount of data to be admitted at each timeslot is given by:

$$R_i(t) = \begin{cases} \min\{L_i(t) + A_i(t), R_i^{\max}\}, & \text{if } \eta Y_i(t) \geq U_i(t), \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The auxiliary variable $\gamma_i(t)$ is calculated at each timeslot. In particular, $\gamma_i(t) = \gamma$, where γ solves

$$\text{Maximize:} \qquad Vg_i(\gamma) - \eta Y_i(t)\gamma \tag{8}$$

$$\text{subject to:} \qquad 0 \leq \gamma \leq R_i^{max} \tag{9}$$

The virtual queues, initially empty, are updated according to:

$$Y_i(t+1) = \max\{Y_i(t) - R_i(t), 0\} + \gamma_i(t) \tag{10}$$

The parameter η satisfies $0 < \eta \leq 1$ and affects a tradeoff between system "learning time" and congestion.

III. Misbehavior pattern

The misbehavior pattern we examine is reporting of a false, larger network queue backlog by one or more nodes. We assume the selfish node constantly reports a backlog greater than k packets than its actual backlog. Our detection approach, described in the next section, is independent of the cheater's strategy, thus we adopt this trivial scenario here, for simplicity.

We performed simulations with MATLAB to evaluate the effect of this behavior in the network. We considered 4 stations, using rounded Gaussian processes for link capacities (i.i.d. across timeslots). Utility functions are logarithmic, $g_i(r_i) = log(r_i)$. The scheduler chooses the station to transmit based on (1). We compare the throughputs and the average backlogs of the nodes at steady state, when all nodes are honest and when the first node constantly reports a larger backlog.

First we examine the case of infinite demand with use of FLOW1. In figure 3 we see what happens when node 1 declares a backlog greater than 10 packets than its true value. We observe a dual performance gain for the cheater. It is increasing its throughput and decreasing its time average congestion. Honest nodes, on the other hand, suffer a deterioration of their performance with decreased throughput and higher congestion.

For the finite demand case we distinguish between two categories of nodes. The first, "non-starving nodes", consists of nodes whose arrival rates coincide with their throughputs, as their demand is fully satisfied. The second, "starving nodes", consists of nodes whose demand is not satisfied fully, their throughput is less than their arrival rate. As the objective of selfish misbehavior we examine is throughput increase, selfish nodes obviously belong to the second category.

We simulated the scenario where nodes 1 and 3 are "starving" and nodes 2 and 4 are "non-starving" and tested the effect

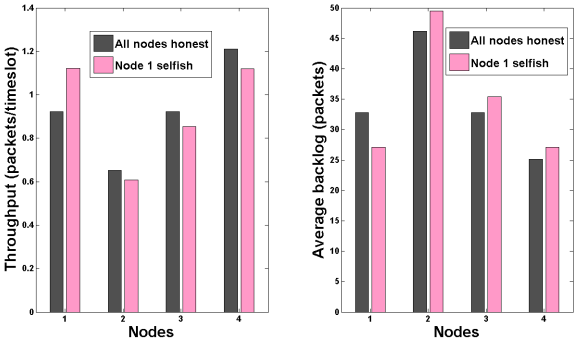

Fig. 3. Infinite demand case: Throughput and average backlog of the four nodes under normal operation and when node 1 misbehaves

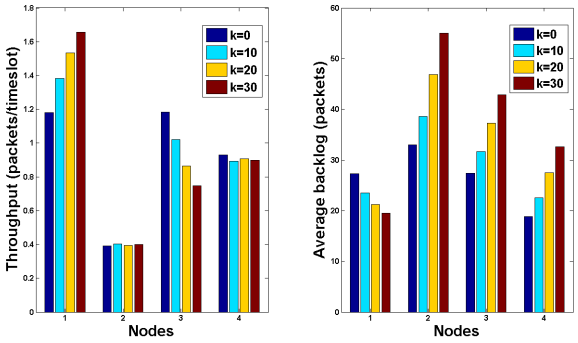

Fig. 4. Throughput and average backlog of the four nodes for $k = 0$(normal operation), $10, 20$ and 30. Nodes 1 and 3 are "starving", nodes 2 and 4 are "non-starving". Node 1 is the misbehaving node.

of node 1 reporting a larger backlog. In figure 4 we see the effect for the cases where $k = 0$ (normal operation), $k = 10$, $k = 20$ and $k = 30$. We observe that "non-starving" nodes experience this behavior as time average delay increase, but they see no drop in throughput. However, further increasing k eventually leads to "non-starving" nodes becoming "starving". Node 3 which is "starving" even in normal operation experiences throughput decrease and congestion increase for any k.

We note here that although we described the misbehavior pattern for an access-point-based single-hop network, this pattern is also applicable to multi-hop networks, where the sources of the flows are potential cheating nodes.

IV. Detection method

For the detection of misreporting behavior, we exploit the structure of the flow control mechanisms of QLA.

A. Infinite traffic demand scenario

Let's first examine the case of infinite traffic demand. Solving the optimization (5) with respect to $R_i(t)$ yields:

$$R_i(t) = \min\{(g')^{-1}(U_i(t)/V), R_i^{max}\} \tag{11}$$

We have

$$\lim_{T\to\infty} 1/T \sum_{\tau=0}^{T-1} R_i(\tau) = \lim_{T\to\infty} 1/T \sum_{\tau=0}^{T-1} \mu_i(\tau) \tag{12}$$

where $\mu_i(\tau)$ is the service rate of node i during timeslot τ. This is a direct consequence of stability. At steady-state, (12) is approximately valid for a large enough interval, so we have:

$$1/T \sum_{\tau=0}^{T-1} R_i(\tau) \approx 1/T \sum_{\tau=0}^{T-1} \mu_i(\tau) \tag{13}$$

Substituting (11) to the approximation above yields

$$1/T \sum_{\tau=0}^{T-1} \mu_i(\tau) \approx 1/T \sum_{\tau=0}^{T-1} \min\{(g')^{-1}(U_i(t)/V), R_i^{max}\} \tag{14}$$

Both hand sides of the above approximation can be calculated at the scheduler side, given that it is aware of the utility functions. Therefore, the detection scheme constitutes in checking the precision of (14) over the interval T during steady state.

An alternative approach for the infinite demand case could be for the access-point to run the cross-layer algorithm by itself, through the use of virtual queue backlogs. Since it is aware of the utility functions, it possesses all the information needed. In this way, virtual flow controllers can be used at the scheduler, and actual flow control can be decoupled from the optimization and substituted by a trivial scheme such as

$$\text{If } U_i(t) < C_i^{max}, \text{ then } R_i(t) = C_i^{max} - U_i(t) \tag{15}$$

This approach is similar to that in [12], where placeholder bits are used to reduce congestion, but different, since now the entire optimization algorithm runs at the scheduler, "virtual admitted packets" can be non-integers and no sophisticated flow control algorithms have to be devised for taking into account fluctuations around the mean backlog levels.

B. Finite traffic demand scenario

Detection in the case of arbitrary arrival rates is not as straightforward. From (10) and (13) we see that at steady-state

$$1/T \sum_{\tau=0}^{T-1} \gamma_i(\tau) \approx 1/T \sum_{\tau=0}^{T-1} R_i(\tau) \approx 1/T \sum_{\tau=0}^{T-1} \mu_i(\tau) \tag{16}$$

At steady-state the constraint (9) can be droppped, so (8) yields

$$\gamma_i(t) = (g')^{-1}(\eta Y_i(t)/V) \tag{17}$$

Comparing (17) with (11) we observe that the "expected" throughput is a function of the virtual queue backlog instead of the actual backlog. However, the nodes report $U_i(t)$ and not $Y_i(t)$. The difference between the virtual and the actual queue backlogs is random and depends on link capacities, arrival rates of "non-starving" nodes and QLA parameters. Thus, we cannot have accurate per timeslot virtual backlog information.

An idea is to use $U_i(t)$ to approximate $\eta Y_i(t)$, that is,

$$\mu_i^{appr} = 1/T \sum_{\tau=0}^{T-1} \min\{(g')^{-1}(U_i(t)/V), R_i^{max}\} \tag{18}$$

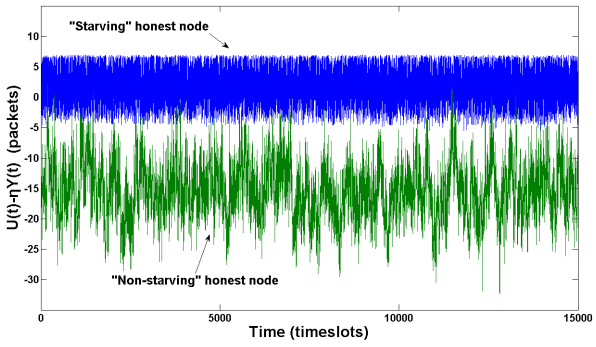

Fig. 5. Evolution of the quantity $U(t) - \eta Y(t)$ for a "starving" and a "non-starving" honest node.

The evolution of $U_i(t) - \eta Y_i(t)$ is depicted in figure 5 for a "starving" honest node and a "non-starving" node. For "non-starving" nodes we see that $U_i(t) - \eta Y_i(t) < 0$. As $g(\cdot)$ is concave, $(g')^{-1}(\cdot)$ is a decreasing function. Thus, (18) yields an overestimation of the throughput. On the other side, for a "starving" node, we see that $U_i(t) - \eta Y_i(t)$ is positive on the average. This is not helpful for detection, since (18) yields an underestimate of the throughput, creating an ambiguity as to if this deviation is normal or due to misbehavior. The deviation is not standard, but varies with channel and arrival statistics, as well as with QLA parameters. A logical approach is to estimate this deviation based on an honest "starving" node, add this to the "expected" throughput and allow for some tolerance. In practice, deviation for selfish users is larger than "normal" deviation, so selfish nodes can be successfully detected.

Let's consider a numerical example. We simulated a network with two "starving" and two "non-starving" stations and assumed that one of the "starving" nodes reports a backlog greater by 5 packets than its true value, while the other is honest. In figure 6 we see their observed throughputs and approximations based on the reported backlogs. We see a difference of 0.07 packets/timeslot for the honest node. Even if we double this and set a threshold of 0.14 packets/timeslot, the selfish node would still be detected as its deviation is 0.29 packets/timeslot. The only case where it avoids detection with this rule is when reporting a backlog larger by only one packet (the simulations showed a deviation of 0.12 packets/timeslot), but then its throughput benefit is marginal (less than 5%). Our observations from simulating various similar scenarios agree that a non-negligible throughput gain always causes a significant deviation between "expected" and observed throughput. Offline training is necessary, to account for the scenario where all starving stations are trying to cheat. Development of more sophisticated methods for the selection of the threshold will be part of our future work.

C. Limitations of the detection approach

An important limitation of the approach is that the scheduling component must be aware of the utility functions of the stations. This can be satisfied if standard utility functions are used or with a priori negotiation of utility functions.

Node	Observed Throughput (Packets/Timeslot)	Approx. based on Backlog (Packets/Timeslot)
Selfish	1.3067	1.0174
Starving honest	1.0969	1.0272

Fig. 6. Numerical example showing difference between "estimated" and "observed" throughputs

Another is with application to multihop networks, as the assumption of a single access point per session ceases to exist. As multiple nodes might be used for the first hop and the link capactities from the source node to them might differ, it is not always possible to keep track of the throughput of the source.

V. Conclusion

We examined the problem of dishonest queue backlog reporting in cross-layer optimization for access-point-based single-hop networks. We showed that a node has a motive to report false backlogs, so as to increase its throughput. Through examination of the flow control components of the quadratic Lyapunov-function based algorithms used with infinite traffic demand and arbitrary input rates, we proposed a simple detection scheme for each case, which assumes knowledge of the utility functions of the stations at the access-point.

Acknowledgement

This work was funded by the research program "CROWN", through the Operational Program "Education and Lifelong Learning 2007-2013" of NSRF, which has been co-financed by EU and Greek national funds.

References

[1] M. J. Neely, E. Modiano and C. P. Li, *Fairness and optimal stochastic control for heterogeneous networks*, in Proc. IEEE Infocom, March 2005.
[2] L. Georgiadis, J. Neely and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*, Monograph, NOW Publishers, 2006.
[3] L. Tassiulas and A. Ephremides, *Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks*, IEEE Trans. Autom. Control, vol.37, No.12, pp.1936-1948, Dec. 1992.
[4] L. Tassiulas and A. Ephremides, *Dynamic server allocation to parallel queues with randomly varying connectivity*, IEEE Transactions on Information Theory, vol.39, No.2, pp.466-478, 1993.
[5] E. H. Clarke, *Multipart pricing of public goods*, Public Choice, 1971.
[6] T. Groves, *Incentives in teams*, Econometrica, pp.617-631, 1973.
[7] F. Kelly, *Charging and Rate Control for Elastic Traffic*, European Transactions on Telecommunications, vol.8, pp.33-37, 1997.
[8] R. Johari and J. N. Tsitsiklis, *Efficiency loss in a network resource allocation game*, Mathematics of Operations Research, vol.29, pp.407-435, March 2004.
[9] G. Hosseinabadi and N. Vaidya, *Selfish misbehavior in scheduling algorithms of wireless networks*, IPCCC 2010:214-221
[10] K. Graffi, P. Mogre, M. Hollick and R. Steinmetz, *Detection of colluding misbehaving nodes in mobile adhoc and wireless mesh networks*, in Proceedings of GLOBECOM 2007, pp.5097-5101.
[11] S. Radosavac, J. S. Baras and I. Koutsopoulos, *A framework for MAC layer misbehavior detection in wireless networks*, Proceedings of ACM Workshop on Wireless Security (WiSe) 2005, Cologne, Germany.
[12] L. Huang and M. J. Neely, *Delay Reduction via Lagrange Multipliers in Stochastic Network Optimization*, Proc. of 7th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), June 2009.