A Proactive Network Coding Strategy for Pervasive Wireless Networking

Elena Fasolo†, Michele Rossi†, Jörg Widmer⋆ and Michele Zorzi†
†DEI, University of Padova, via Gradenigo 6/B – 35131 Padova, Italy
⋆DoCoMo Euro-Labs, Landsberger Strasse 312 – 80687 Munich, Germany

***Abstract*— In recent years, network coding has proved to be an efficient tool to disseminate data through a network. A number of practical schemes have been proposed to implement network coding also in wireless environments. Most of them are based on reactive and probabilistic random network coding and their effectiveness has been investigated under the assumption of idealized network conditions. However, recent work has shown that the benefits of such strategies decrease when applied in realistic network environments. In this paper, we propose an algorithm to efficiently disseminate data through network coding in realistic wireless networks by using a proactive approach, named *ProNC*. We develop a distributed and self-adaptable protocol which substantially increases the performance of network coding in practical scenarios and achieves full reliability with both low protocol overhead and low delay. We show the effectiveness of *ProNC* via ns-2 simulations and compare it with previously proposed schemes.**

I. Introduction

One of the challenging problems in wireless pervasive systems is the efficient dissemination and collection of data spread over a distributed network [1]. Classic schemes dealing with this problem are based on the *store and forward* paradigm (see flooding, epidemic and probabilistic routing [2], [3]) where nodes store new incoming information and forward it, whenever possible, to other nodes. Their main disadvantage is the high overhead required for information dissemination.

Algorithms based on network coding allow to more efficiently deliver data through such a network [4]. These schemes implement a *store, code, and forward* paradigm where information packets are encoded at intermediate nodes and subsequently forwarded. Coding improves the dissemination efficiency by reducing the number of transmissions required per successfully delivered information unit.

A number of practical schemes exploiting this paradigm have been proposed. In general, they are based on *random linear network coding* which is well suited for distributed networks. Packets flowing through the network contain linear combinations of *original* packets X_m inserted into the network by some sources nodes: $Y_i = \sum_m g_{i,m} X_m$. The coding coefficients $g_{i,m}$ are picked randomly from a finite field of fixed size. Received information is stored in a buffer and at transmission time, a node simply sends out a random linear combination of the (coded) packets it has stored.

To retrieve the M original packets, a node has to have M linearly independent packets in its buffer and thus has to solve a system of M linear equations (by inverting the matrix of coding coefficients $X = G^{-1}Y$, e.g., through Gaussian elimination). This requires knowledge of the coding coefficients, which, for example, can be included in the header of the coded packets. A more detailed description can be found in [5], [6]. Similar to erasure codes, it is not important which linear combinations a node receives, as long as it receives a sufficient number of linearly independent ones. Furthermore, together with physical layer broadcast, a coded packet is likely to bring new information to many of the nodes overhearing it.

An actual protocol requires a mechanism to determine at what time and how many coded packets are to be sent. Probabilistic random network coding schemes based on a *forwarding factor* parameter have been presented in [7], [8]. The forwarding factor ρ is defined as the average number of packets that each node transmits divided by the number of innovative packets received and there are different ways to select it. It can be fixed and equal for all nodes (i.e., *reactive network coding with fixed* ρ); or it can depend on the number of neighbors $n_v(x)$ of the transmitter x (e.g., $\rho = k/n_v(x)$ for some $k \geq 1$). The latter approach is based on the intuition that a certain number of transmissions are required in each neighborhood to allow decoding at all nodes. The higher the node density, the lower the number of transmissions to perform at each individual node. The effectiveness of these schemes is analyzed in [7], [8], where they are evaluated under ideal MAC conditions. Recent studies [9], however, have shown that in case of all-to-all communications, the benefits of existing random network coding strategies can drastically decrease when they are applied to actual wireless CSMA/CA based networks. In particular, their performance suffers due to the presence of both collisions and random access scheduling. Finally, the practical applicability of network coding for unicast communications is studied in [10].

All the network coding schemes introduced previously are based on a *reactive* approach. In this paper, instead, we propose a network coding data dissemination scheme based on a *proactive* approach (referred to in the following as *ProNC*) which achieves good performance also in actual CSMA/CA environments. In particular, we focus on scenarios where data is to be exchanged among all the users of a wireless ad hoc network. Our scheme is completely distributed and self-adaptable and requires very limited network knowledge, which can be easily acquired by overhearing the exchanged data. We show the superiority of our approach by comparing it against existing network coding strategies [7] and against an idealized scheme with a perfect priority scheduler. In the latter case, access priorities are calculated using full knowledge of the buffer contents of all nodes.

The details of the proactive network coding strategy are given in Section II, and in Section III we show the effectiveness of the algorithm through extensive simulations with ns-2. Section IV concludes the paper.

1930-529X/07/$25.00 © 2007 IEEE

II. Proactive Network Coding (ProNC)

A challenging problem in wireless ad hoc networks consists of efficiently disseminating information network-wide. In this paper, we address this problem by devising broadcast schemes based on network coding. We consider a scenario where at every node there is an application which inserts packets into the network and where all nodes want to collect all the inserted packets. We refer to the inserted packets as *original packets*. Probabilistic reactive network coding is generally a good solution for broadcasting data in these settings [7]; however, previous work [9] has highlighted that this technique is likely to suffer from the presence of interference and collisions in actual radio environments. The main problem of reactive schemes is that new random combinations of packets are only generated and transmitted when innovative (i.e., linearly independent) information is received. Innovative packets may however be lost in scenarios with packet collisions, thus interrupting data propagation. Furthermore, insertion of innovative information into an area often causes all nodes in the area to attempt their new transmissions simultaneously and this further increases the collision probability.

Also, in reactive probabilistic network coding, nodes send out new combinations based on a forwarding factor ρ which depends on the number of neighbors they have [7]. We observe that there are particular topologies where this strategy does not work. As an example, think of the case where a given node x has a large number of neighbors and one of them, say node y, has only x as its neighbor. Due to its high number of neighbors (small ρ), x sends out a small number of packets and, in turn, y is unlikely to be able to decode all the wanted information (as it did not receive enough independent combinations from x). We may alternatively increase ρ until the number of transmissions per node allows the recovery of a sufficient number of packets also in the above case. Unfortunately this solution has two drawbacks: increasing ρ leads to more severe channel congestion and increased overhead (number of packets sent per original packet). This is clearly not desirable as it largely neutralizes the performance gain due to network coding. Another possible solution is presented in [1]. This scheme is based on a push and pull mechanism where, at the end of the data dissemination phase, each node that is not able to decode asks other nodes for the missing packets.

In this paper we look at an alternative strategy. Instead of considering the reactive paradigm introduced above, we adopt a *proactive* approach where each node periodically sends out new random packet combinations. The main advantages of a proactive scheme are: 1) it does not require the reception of innovative information to continue data dissemination, so it is more robust to interference and collisions; and 2) its performance does not depend on the specific choice of the forwarding factor ρ. A proactive data dissemination scheme needs two important components to work:

1) A set of conditions to stop transmissions when all original packets have been delivered to all nodes, i.e., *Stopping Conditions (SC)*.
2) A strategy to set the frequency at which the new random packet combinations are sent so as to avoid network congestion and to save energy consumption. In the rest of the paper we refer to this strategy as *Rate Adaptation* mechanism.

In Section II-A, we first describe the basic rules of our proactive network coding (*ProNC*) data dissemination scheme. In Section II-B, we define the *Stopping Conditions* and, in Section II-C, we discuss the problem of finding a proper *Rate Adaptation* heuristic. Finally, in Section II-D, we highlight some aspects related to the implementation of *ProNC*.

A. Basic Rules for ProNC

We assume that each node can be in one of two different states: *active* and *inactive*. The basic idea of the proactive approach is that an *active* node periodically sends out a new packet combination to its neighbors, while an *inactive* node does not transmit. To switch from one state to the other, a node considers the following set of rules:

Rule 1: *A node becomes active upon receiving the first innovative packet.* This means that a data dissemination phase is started and the node has to contribute to it.

Rule 2: *A node becomes inactive when the Stopping Condition is verified.* In this case, further transmissions from this node are no longer useful for its neighbors and should be suppressed to avoid unnecessary overhead.

Rule 3: *A node becomes active again when the Stopping Condition no longer holds.* This last rule is particulary important as it allows propagation of new information into an area where all nodes are currently inactive.

Note that while a node is inactive, it can still receive packets from its neighbors. This information shall be used to assess whether the stopping condition still holds.

B. Stopping Conditions

There are different ways to define the *Stopping Conditions* for proactive network coding. They depend, in general, on the amount of information that each node has to collect in order to decide whether to suspend its transmissions. Our main aim is to keep the overhead as low as possible. The motivations for this are twofold. Our scheme should be distributed and self-adaptable so that nodes should not need full knowledge about the network topology/status. Moreover, maintaining the overhead low contributes to avoiding network congestion for a given traffic condition.

We identify two simple cases in which a node has to suspend its transmission. In the first case, all neighbors of a node x have decoded all the packets they require and thus no further transmissions by x are necessary. The second is when the subspace spanned by the information vectors (i.e., packets) available at node x is contained in the subspace spanned by the information vectors at each of the node's neighbors. In this case, x's packets will not be innovative for any of its neighbors and the node should suspend its transmission.

Based on these observations, we propose two different conditions which are referred to as *Strong* and *Weak* Stopping Conditions (*SSC* and *WSC*, respectively). They implicitly define two different proactive schemes. According to the *SSC*, nodes send out beacons (*Strong Stopping Messages, SSM*) to their neighbors when they have decoded all the packets they are

1930-529X/07/$25.00 © 2007 IEEE

interested in. Each node collects *SSM*s from its neighborhood in order to autonomously verify the *SSC* and thus suspend its own transmission. We refer to this scheme as *Strong ProNC* as it requires strong assumptions on the data traffic. In particular, each node, in order to send out *SSM*s, needs to know in advance how many packets it wants to collect. This fact implies that each node has full knowledge about the amount (and type) of data flowing over the network. Note that the collection of this information, in practice, may be infeasible. As a practical example, imagine that a node is interested in collecting sensor readings from all sensors placed in a specific area. In order to send a *SSM*, the node must know in advance the number of transmitting sensors and hence the number of packets to collect.

According to the *WSC* scheme, each node suspends its transmissions when all its neighbors can decode all the packets in their buffers and their decoding matrices all have the same rank. To verify this condition, during data propagation, each node sends out beacons (*Weak Stopping Messages, WSM*) containing a *decoding field* which is set to 1 if it can decode all packets in its buffer and to 0 otherwise. In addition, beacons contain a *rank field* specifying the rank of the nodes' decoding matrices. We refer to this second strategy as *Weak ProNC* because it does not require any knowledge about the data traffic and has a limited overhead. However, *Weak ProNC* is suboptimal as there are some situations in which the rank alone does not capture the exact decoding status at different nodes. For instance, it might happen that all neighbors of a node can decode all the packets in their buffers and they all have the same rank but the decoded information is different. We compare the performance of both *Strong* and *Weak ProNC* in Section III. In addition, we also consider, as a benchmark, a scheme based on the complete knowledge of all the buffers' contents and where an omniscient entity regulates the packet transmissions. Note that implementing such a scheme in a distributed network requires to have each node send out state vectors describing the content of its own buffer at the reception of an innovative packet (in a way similar to [10]). In case of all–to–all communications this leads to an unsustainable overhead in terms of transmissions and memory.

C. Rate Adaptation Heuristic

When a node is active, it periodically sends out new random packet combinations generated from its own buffer. A crucial aspect of *ProNC* is therefore the adopted rate adaptation strategy, as previously defined in Section II. A proper selection of the transmission rate used at each node is important to avoid congestion while achieving good decoding performance. This selection translates into choosing a proper time interval between two consecutive transmissions, referred to here as τ.

Our first aim is to avoid both synchronization among nodes and a high simultaneous usage of the channel. For this reason, we take τ as a uniform random variable in $[\tau_{avg} - \tau_{avg}/2, \tau_{avg} + \tau_{avg}/2]$, where τ_{avg} is the average value of τ. We also define the quantity $\mu_{avg} = 1/\tau_{avg}$ which represents the average packet transmission rate. In addition, we allow τ to vary across consecutive transmissions and across nodes. This avoids synchronization and limits channel contention. The second problem to be considered is the selection of a good value for μ_{avg}; when all nodes use a high transmission rate, channel collisions are the bottleneck and, in turn, we would expect unsatisfactory delay performance. However, at very low transmission rates the delivery delay will also be very long and information will slowly propagate through the network. For intermediate transmission rates, an optimal value of μ_{avg} should exist. This optimal value should minimize the delay while giving acceptable protocol overhead performance (packet transmissions per recovered packet). We finally note that the optimal μ_{avg} should depend on the node density (number of neighbors per node). In particular, for increasing densities μ_{avg} should decrease so as to keep the number of collisions at an acceptable level.

In Section III-B we analyze by means of simulations, that an optimal transmission rate in fact exists and we validate the statement above. Based on these results, in Section III-C we detail a density dependent rate adaptation heuristic and we compare its performance against reactive network coding.

D. Implementation Notes

ProNC requires the estimation of the number of neighbors at each node. This can be simply achieved by monitoring the source addresses of incoming packets. Note that both the stopping conditions and the rate adaptation mechanism depend on the node density. In addition, *Stopping Messages* are included within data packets at the cost of a few extra bits. For *SSM*, we need one additional bit, whereas for *WSM* we need a bit to represent the decoding status and a byte to communicate the rank of the local decoding matrix[1]. In both cases, the additional overhead is acceptable. On the downside, when a node becomes inactive it must send out at least one *Stopping Message* to communicate its change of status and this packet may be useless for coding purposes.

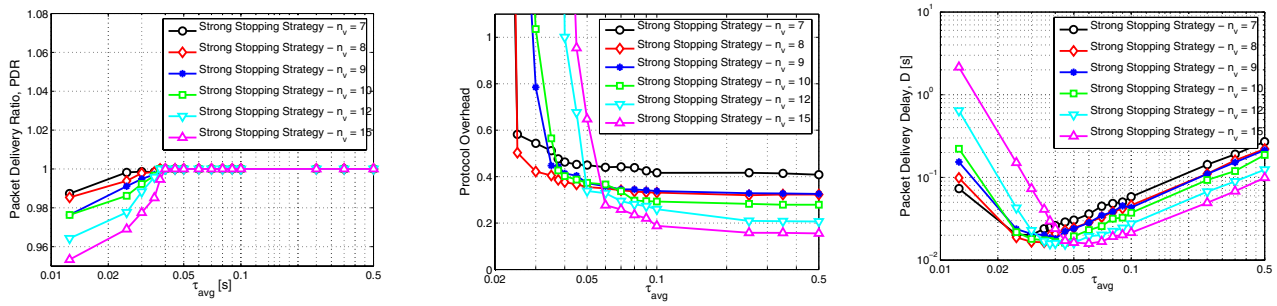
We stress that piggybacking control information within data packets has the beneficial effect of keeping channel congestion low. In addition, the added control information (*SSM*s and *WSM*s, rank, decoding status) is used to increase the efficiency of network coding schemes which, in turn, can further reduce the number of transmissions for a target performance level. These benefits are quantitatively verified in Section III.

III. Simulation Results

In this section, we evaluate the performance of *ProNC* by means of ns-2 simulations. First of all, in Section III-A, we briefly introduce network topology and traffic pattern and define the considered performance metrics. In Section III-B, we analyze the behavior of *Strong* and *Weak ProNC* for different transmission rates. Subsequently, in Section III-C we introduce and evaluate a rate adaptation heuristic. Finally, in Section III-D, we compare *ProNC* against the reactive probabilistic network coding schemes proposed in [7], [9].

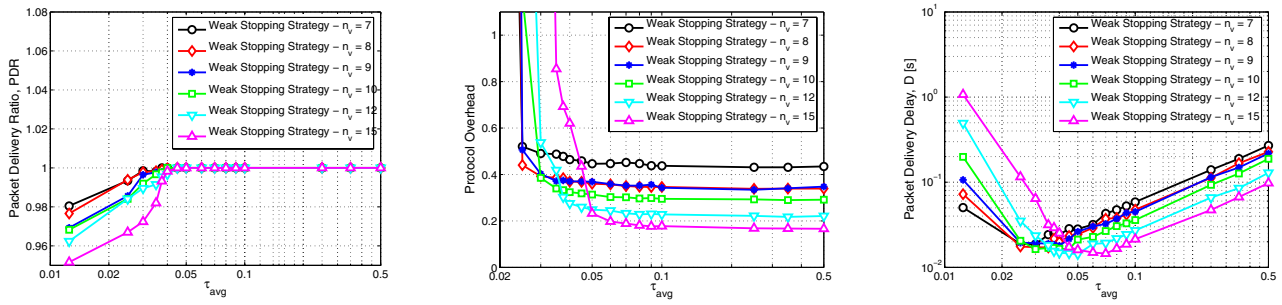
[1]A single byte often suffices in practice, i.e., when the number of packets to be coded together is lower than or equal to 256. Coding over more original packets would imply the inversion, at the receivers, of large matrices which is impractical and difficult to obtain as a realtime operation.

1930-529X/07/$25.00 © 2007 IEEE



(a) Packet Delivery Ratio (b) Protocol Overhead (c) Packet Delivery Delay

Fig. 1. Performance of *Strong ProNC* as a function of the average insertion interval, τ_{avg}.



(a) Packet Delivery Ratio (b) Protocol Overhead (c) Packet Delivery Delay

Fig. 2. Performance of *Weak ProNC* as a function of the average insertion interval, τ_{avg}.

A. Reference Scenario and Performance Metrics

We consider random topologies as they capture well the main characteristics of actual network settings, especially in wireless ad hoc scenarios. Nodes are randomly placed within a fixed area in such a way that the topology is always connected but the paths among sources and destinations can be multi-hop. We consider several average node densities by varying the average number of neighbors, $n_v \in \{7, 8, 9, 10, 12, 15\}$. For medium access control, we adopt the basic IEEE802.11b broadcast mode, accounting for channel errors and collisions. In addition, we assume that at the beginning of the simulation each node has a single original packet to disseminate to all other nodes.

Next, we define the performance metrics that will be used, in Section III-B, to study the behavior of *ProNC*:

Packet Delivery Ratio, PDR: is defined as the ratio between the number of successfully received (and decoded) packets, and the total number of packets a node is interested in. This metric is averaged over all nodes.

Packet Delivery Delay, D: is the time between the insertion of an original packet (i.e., the beginning of the simulation) and its successful decoding at a receiver, averaged over all nodes that receive it and over all the original packets.

Protocol Overhead: is defined as the ratio between the total number of packets transmitted at the MAC layer (including also control packets) and the total number of packets successfully decoded, summed over all nodes[2].

[2]Note that, due to the broadcast nature of the channel and the use of network coding, this ratio could be less than 1.

B. Evaluation of Strong and Weak Stopping Policies

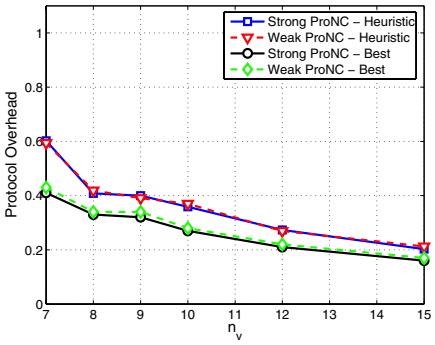
In this section we study the behavior of *ProNC* under different network conditions when varying $\tau_{avg} = 1/\mu_{avg}$. The following results are plotted as a function of τ_{avg} to emphasize the impact of inter-packet transmission times on network coding performance. In this section we assume τ_{avg} as a fixed parameter, equal for all nodes and independent of the node density.

In Figs. 1 and 2, we show the performance of *Strong ProNC* and *Weak ProNC*, respectively. The behavior of these strategies is very similar, especially for high values of τ_{avg} (i.e, low transmission rates). In Fig. 1(a) and Fig. 2(a) we plot the packet delivery ratio. This metric is always equal to one except for the cases where τ_{avg} is very small (high μ_{avg}). Under these operating conditions the protocol overhead performance is considerably impacted as well, as can be observed from Figs. 1(b) and 2(b). These facts are due to the high collision rates that we get when τ_{avg} is excessively small. We further observe that, when the network is congested (small τ_{avg}), *Weak ProNC* performs slightly better than *Strong ProNC*. In fact, *Weak Stopping Conditions* allow nodes to momentarily stop their transmissions when their packets are unlikely to be innovative for their respective neighbors. However, as soon as new information arrives, the nodes will go back to the active state. Note that this requires a continuous assessment of the *Stopping Condition*. Under a *Strong ProNC* this may not happen as the nodes remain in the active state until they receive a *SSM* from *all* their neighbors. However, packet collisions may prevent the reception of *SSM*s from all neighbors and in this case a node will unnecessarily continue transmitting.

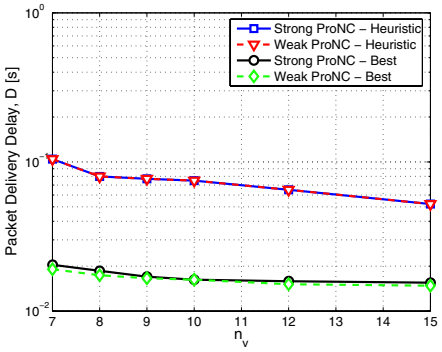
1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

	Strong ProNC		*Weak ProNC*	
n_v	τ [s]	α	τ [s]	α
7	0.030	0.0037	0.030	0.0037
8	0.035	0.0039	0.035	0.0039
9	0.040	0.0040	0.040	0.0040
10	0.045	0.0041	0.045	0.0041
12	0.050	0.0038	0.060	0.0046
15	0.060	0.0037	0.070	0.0044

(a) Optimal τ_{avg} and α



(b) Protocol Overhead



(c) Packet Delivery Delay

Fig. 3. *Weak* and *Strong ProNC*: comparison of the performance of the rate adaptation heuristic with the best achievable.

In Figs. 1(c) and 2(c) we report the packet delivery delay. As expected, we observe that the delay curves have a minimum for specific values of τ_{avg}. As discussed earlier, for small τ_{avg} the delay increases due to the severe channel congestion. When τ_{avg} is large, instead, the delay increases due to the long lapse of time between consecutive transmissions. We additionally observe that the values of τ_{avg} minimizing the delay are slightly different for different node densities (n_v in the figures) and they do not always coincide with the values minimizing the protocol overhead. We finally note that the differences between *Weak* and *Strong* strategies in terms of minimum delays and optimal τ_{avg} are very small.

C. A Possible Rate Adaptation Heuristic

As we mentioned in Section II-C, due to the proactive nature of our scheme, we need to define a proper rate adaptation heuristic to guarantee good performance while making the data dissemination scheme dynamic and self-adaptable. The results in Section III-B show that an optimal value of τ_{avg} exists and that it depends on the node density. Fig. 3(a) reports the optimal τ_{avg} for different nodes densities. As done in [7], we assume a linear relationship between number of neighbors and transmission rate at any given node[3]. Accordingly, τ_{avg} can be expressed as:

$$\tau_{avg} = \alpha(n_v + 1)\,, \tag{1}$$

where α is a constant, n_v is the average number of neighbors per node and $n_v + 1$ the average number of nodes contending for the channel in a given neighborhood. Now, considering the optimal τ_{avg}, which can be found by simulations as illustrated in the previous section, we derive α as:

$$\alpha = \frac{\tau_{avg}}{n_v + 1}\,. \tag{2}$$

In Fig. 3(a) we report the obtained values of α. Notably, these values are very close for different node densities. Thus, to define our heuristic, we considered $\alpha_{avg} = 0.004$, obtained by averaging α over all densities. Hence, during the dissemination phase, each node x calculates its $\tau(x)$ as: $\tau(x) = \frac{\alpha_{avg}}{n_v(x)+1}$, where $n_v(x)$ is the actual number of neighbors of node x. Note that the choice of τ is approximated but it can be derived in a distributed way (each node only requires a local estimation of the number of its own neighbors) and is allowed to be different for different nodes.

We evaluate the performance of the above *Rate Adaptation* heuristic in Figs. 3(b) and 3(c). We omit the packet delivery ratio as it is always equal to one. For comparison, in these figures we also plot two additional curves, for *Weak* and *Strong Stopping* policies, which are obtained by calculating the best performance in Figs. 1 and 2. From Fig. 3(b) we observe that the proposed heuristic leads to a small degradation with respect to the overall minimum overhead obtained by choosing the optimal value of α for each node. From Fig. 3(c), we note that the gap with respect to the minimum achievable delay is larger. We finally observe that the degradation incurred in adopting our heuristic in place of a fixed τ_{avg} is the price to pay to have a fully distributed and self-tunable scheme. In fact, if we used a fixed τ_{avg}, we would have to know the average node density in advance to achieve optimal performance. However, this knowledge may be hard to obtain in practice. With our heuristic, the performance is slightly decreased but we gain in generality as the resulting scheme works for any topology and without any knowledge about the node density.

To summarize, we can state that both *Strong* and *Weak ProNC* show satisfactory performance in actual network settings. In particular, *Weak ProNC* with our rate adaptation heuristic is a completely distributed and self-adaptable algorithm. Moreover, it does not require any knowledge about the traffic and only requires a few local interactions among nodes to work properly.

D. ProNC vs Reactive Network Coding Schemes

In this section, we compare the *ProNC* scheme (with our rate adaptation heuristic) against the reactive probabilistic schemes proposed in [7] and a scheme based on complete knowledge about the network status, referred to as *Innovation-based Network Coding, (INC)*. This last scheme is introduced here to get an upper bound on achievable performance.

In Innovation-based Network Coding nodes gain the opportunity to transmit new packets according to their priority, which is determined by assuming to have a *complete knowledge of the network status*. The transmission priority accounts for the innovative contribution that each packet transmission from a given node x could bring to x's neighbors. The higher the

[3]In [7] the forwarding factor ρ implicitly defines the transmission rate at each node by modulating the transmission process.

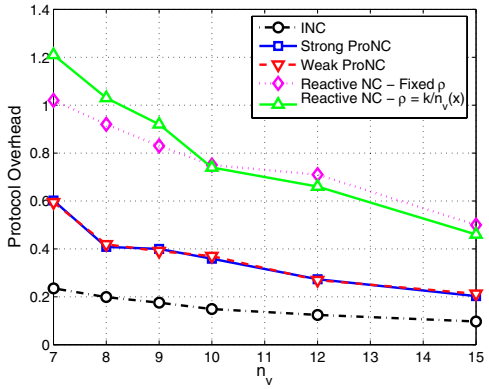


Fig. 4. *Weak* and *Strong ProNC* against *INC* and reactive schemes

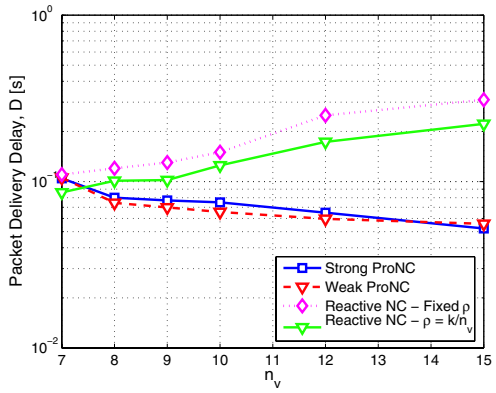


Fig. 5. *Weak* and *Strong ProNC* against reactive schemes

innovative contribution of node x, the higher its transmission priority. Let $S_x(t)$ be the set of all coded messages in node x's buffer at time t. We define the transmission priority $P_{TX}(x,t)$ of node x at time t as follows:

$$P_{TX}(x,t) = \sum_{i=1}^{n_v(x,t)} \dim(S_x(t) \setminus S_i(t)), \qquad (3)$$

where $n_v(x,t)$ is the number of neighbors of node x at time t, $S_x(t)$ and $S_i(t)$ are the subspaces spanned by the code vectors at nodes x and i at time t, respectively, and $\dim(\cdot)$ returns the dimension of a subspace. $B \setminus A$ returns the set $\{x \in B \text{ and } x \notin A\}$, whereas the index i spans over x's neighbors. At time t, all nodes with the highest (non zero) priority transmit. Each transmission is assumed to be error free. Transmitters generate a new combination from their buffers and send it to their neighbors. At this point, the new network status is computed, i.e., priorities are updated, and a new transmission round (time $t+1$) starts. The procedure continues until all nodes decode all packets (i.e., all priorities are equal to zero). This scheme always guarantees a packet delivery ratio of one and distributes the innovative information as fast as possible. Note that this is an idealized scheme which is considered here as an upper bound on *ProNC* performance.

In Fig. 4, we compare *Strong* and *Weak ProNC* against *INC* and reactive network coding with fixed and adaptive forwarding factor (see [7]). We only report the protocol overhead as we compare the schemes for the same value of the packet delivery ratio, i.e., PDR$= 1$. We observe that *ProNC* performs closely to *INC*. Hence, having full knowledge of the network only gives marginal improvement in the considered cases. However, achieving this knowledge in practice may require the transmission of a substantial amount of control traffic, which may drastically reduce the benefits of *INC*. In addition, *ProNC* obtains substantial gains over reactive schemes in terms of protocol overhead (the overhead is roughly halved with *ProNC*). Note that, in this scenario, other schemes such as the probabilistic flooding can achieve a maximum $PDR = 0.9$ due to the collisions leading to a protocol overhead around 2 [9].

Finally, in Fig. 5 we report the delivery delay performance. At low densities all schemes give similar results, whereas as the number of neighbors increases *ProNC* guarantees a packet delivery delay that is almost one order of magnitude smaller than that of reactive solutions. In addition, it is observed that with *ProNC* channel congestion is successfully mitigated for a wide range of node densities. This is possible thanks to the adaptation carried out by our heuristic. Reactive schemes, instead, heavily suffer from an increasing density, which ultimately leads to long delays.

IV. Discussions and Conclusions

In this paper, we presented an original network coding scheme for data dissemination. In contrast to prior work, it exploits a proactive approach (*ProNC*), which solves some of the problems of network coding in realistic wireless environments. Our algorithm is distributed and self-adaptable. Also, the scheme requires some local coordination among nodes, which can be achieved through piggybacking control messages at a reasonable overhead. We evaluated the effectiveness of *ProNC* in distributed wireless settings, getting very good performance for all considered cases. Some issues are still open such as the evaluation/adaptation of our scheme in multicast/unicast scenarios with non-homogeneous traffic.

References

[1] E. Fasolo, C. Prehofer, M. Rossi, Q. Wei, J. Widmer, A. Zanella, and M. Zorzi, "Challenges and new approaches for efficient data gathering and dissemination in pervasive wireless networks," in *INTERSENSE*, Nice, France, Apr. 2006.

[2] P. T. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulié, "Epidemic information dissemination in distributed system," *IEEE Computer Magazine*, vol. 37, no. 5, pp. 60–67, May 2004.

[3] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, July 2003.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. on Information Theory*, vol. 46, no. 4, July 2000.

[5] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. ISIT*, June 2003.

[6] P. A. Chou, T. Wu, and K. Jain, "Practical network coding," in *41st Allerton Conf. Communication, Control and Computing*, Monticello, IL, US, Oct. 2003.

[7] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "A network coding approach to energy efficient broadcasting: from theory to practice," in *IEEE Infocom*, Barcelona, Spain, Apr. 2006.

[8] J. Widmer, C. Fragouli, and J.-Y. L. Boudec, "Low–complexity energy–efficient broadcasting in wireless ad–hoc networks usign network coding," in *NetCod*, Riva del Garda, Italy, Apr. 2005.

[9] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "On MAC Scheduling and Packet Combination Strategies for Random Network Coding," in *IEEE ICC 2007*, Glasgow, UK, Jun. 2007.

[10] S. Katti, H. Rahul, W. Huss, D. Katabi, M. Medard, and J. Crowcroft, "XORs in The Air: Practical Wireless Network Coding," in *ACM SIGCOMM*, Pisa, Italy, Sep. 2006.