Mixer Notes

March 5, 2019

1 Discovery Phase: Tx Probability and Exit

1.1 Definitions

$N \ldots$ number of nodes in the network
$p_T(k) \ldots$ probability that a node decides to transmit a packet in slot k

1.2 Model

1.2.1 Assumptions

The model is based on the following key assumption: For each node i in the network: *If multiple neighbors of node i transmit at the same time, then node i receives the packet from its strongest neighbor.* The word "strongest" refers to some physical quantity that approximately fits the capture behavior (e.g. Rx power). Let $N_i = \{1, \ldots, N\} \setminus i$ the set of neighbors[1] of node i. W.l.o.g. we sort N_i by "strength" in decreasing order (from node i's perspective), i.e., index 1 belongs to i's strongest neighbor.

Further, we assume that each node in the network transmits in slot k with probability $p_T(k)$. The transmit decision is made independent of other nodes and independent from all previous slots. As a consequence, the transmit decision of all nodes in slot k is i.i.d..

Remarks. The key assumption does not exactly match the capture behavior of real receivers. One important quantity to understand capture is SINR. SINR depends on the whole set of transmitting nodes, not just the strongest one. It is easy to see that the assumption will not reliably reproduce the capture behavior if packets from two transmitters arrive with almost the same power. However, we assume that the model is an acceptable approximation for many cases.

1.2.2 Rx Probability

Consider node i with neighborhood N_i. Let $p_n(k)$ the probability that neighbor n (indexing as defined above, i.e., n'th element of N_i) is received by node i in

[1] Regarding the model, i.e., without additional restrictions, the neighborhood of a node is the whole network. In practice it is meaningful to delimit the neighborhood based on Rx power.

slot k. With our key assumption it holds

$$\begin{aligned} p_n(k) &= \underbrace{p_T(k)}_{\text{node } n} \cdot \underbrace{(1-p_T(k))^{n-1}}_{\text{nodes } 1,\ldots,n-1} \cdot \underbrace{(1-p_T(k))}_{\text{node } i} \\ &= p_T(k) \cdot (1-p_T(k))^n \end{aligned} \tag{1}$$

Node n will be received by node i if it transmits something and if all stronger neighbors do not transmit and – not to forget – if node i is in receive mode (the latter condition can be kept implicitly by inserting i into N_i as element 0).

1.2.3 Discovery Probability

Let $p_D(n)$ the probability that neighbor n has been discovered by node i in slot K, i.e., it has been received at least once during slots $1 \ldots K$. It holds

$$p_D(n) = 1 - \prod_{k=1}^{K} \big(1 - p_n(k)\big) = 1 - \prod_{k=1}^{K} \big(1 - p_T(k)(1-p_T(k))^n\big) \tag{2}$$

Note that p_D is monotonically increasing in K and decreasing in n.

1.3 Maximizing Discovery Probability

Goal: Choose $p_T(k)$ such that $p_D \to \max$.

1.3.1 Discovery Process

During discovery phase, the set $\hat{N}_i \subseteq N_i$ of known neighbors forms an increasing set. Before the first slot, node i knows only itself. After the first reception, it knows one neighbor. After n receptions it knows at most n neighbors. Note that for the task of discovery it does not matter how strong the detected neighbors are compared to unknown neighbors.

For designing the discovery phase (choosing parameters and so on), the most interesting question is: Given a subset of detected neighbors $\hat{N}_i$ in slot K, what is the probability that there are more (undetected) neighbors? Besides the dependency on K, this probability is highest if $\tilde{N}_i$ covers the first $|\tilde{N}_i|$ entries of N_i (the strongest neighbors), i.e., it is upper bounded[2] by $1 - p_D(|\tilde{N}_i| + 1)$. Hence, we focus on maximizing p_D.

1.3.2 Objective Function

We target on discovery settings that are universal (as much as possible), i.e., that work satisfactory for a varying number of neighbors. Therefore we use a weighted sum over multiple values of $|N_i|$, ranging from 1 to N. This leads to the following objective

$$\max_{p_T \in \mathbb{R}^K} \sum_{n=1}^{N} g_n \, p_D(n) \quad \to \quad \min_{p_T \in \mathbb{R}^K} \sum_{n=1}^{N} \left[g_n \prod_{k=1}^{K} \Big(1 - p_T(k)(1-p_T(k))^n\Big) \right] \tag{3}$$

[2] strictly speaking this formulation is wrong (upper bound depends on the joint probability from all remaining nodes), but the basic statement (focus on p_D) is valid

1.3.3 Optimization

Rewriting the product term in (3) as

$$\exp\left(\ln g_n + \sum_{k=1}^{K} \ln\left(1 - p_T(k)(1-p_T(k))^n\right)\right) \tag{4}$$

unveils that the variables $p_T(k)$ can be decoupled because $\exp(\cdot)$ and $\ln(\cdot)$ are strictly monotonic functions. Hence, we build the partial derivatives:

$$\begin{aligned}
\frac{\partial}{\partial p_T[\tilde{k}]}(3) &= \frac{\partial}{\partial p_T[\tilde{k}]} \sum_{n=1}^{N} \left[g_n \prod_{k\neq\tilde{k}} \Big(\underbrace{1 - p_T(k)(1-p_T(k))^n}_{(*)} \Big) \Big(1 - p_T(\tilde{k})(1-p_T(\tilde{k}))^n\Big) \right] \\
&= \frac{\partial}{\partial p_T[\tilde{k}]} \left[\sum_{n=1}^{N} g_n \prod_{k\neq\tilde{k}} (*) - \sum_{n=1}^{N} g_n \prod_{k\neq\tilde{k}} (*)\; p_T(\tilde{k})(1-p_T(\tilde{k}))^n \right] \\
&= -\sum_{n=1}^{N} g_n \prod_{k\neq\tilde{k}} (*) \left((1-p_T(\tilde{k}))^n - n p_T(\tilde{k})(1-p_T(\tilde{k}))^{n-1} \right) \\
&= -\sum_{n=1}^{N} g_n \prod_{k\neq\tilde{k}} (*) \left(1-p_T(\tilde{k})\right)^{n-1} \left(1 - p_T(\tilde{k}) - n p_T(\tilde{k})\right) \\
&= -\sum_{n=1}^{N} g_n \prod_{k\neq\tilde{k}} (*) \left(1-p_T(\tilde{k})\right)^{n-1} \left(1 - (n+1)\, p_T(\tilde{k})\right)
\end{aligned} \tag{5}$$

Set to zero and split:

$$\sum_{n=1}^{N} g_n \prod_{k\neq\tilde{k}} (*) \left(1-p_T(\tilde{k})\right)^{n-1} = p_T(\tilde{k}) \sum_{n=1}^{N} g_n \prod_{k\neq\tilde{k}} (*) \left(1-p_T(\tilde{k})\right)^{n-1} (n+1)$$

$$p_T(\tilde{k}) = \frac{\sum_{n=1}^{N} \left[g_n \prod_{k\neq\tilde{k}} (1 - p_T(k)(1-p_T(k))^n) \left(1-p_T(\tilde{k})\right)^{n-1} \right]}{\sum_{n=1}^{N} \left[g_n \prod_{k\neq\tilde{k}} (1 - p_T(k)(1-p_T(k))^n) \left(1-p_T(\tilde{k})\right)^{n-1} (n+1) \right]} \tag{6}$$

While (6) appears non-trivial, it has an interesting property: Due to its form $\sum_n a_n / \sum_n (a_n(n+1))$ it describes a contraction mapping (Lipschitz constant $L < 1$). Hence, the function has one fixed-point, and we can use a fixed-point iteration to converge to that point (Banach fixed-point theorem).

TODO:

- describe why $p_T(1) = p_T(2) = \ldots = p_T(K)$ in the fixed-point
- analyze if that value can be derived analytically
- investigate other things like convexity, KKT conditions, and so on

1.4 Applying Results to Mixer

First note that (2) can be used to compute discovery probabilities resulting from arbitrary p_T. Hence, it can be used to check the effect of fine-tuning p_T as described next.

1.4.1 Estimating Network Density

The weights g_n in (3) can be used to damp configurations that seem improbable. One typical usage is to suppress configurations with unrealistic network densities ($|N_i|$). We choose $|N_i|$ depending on the number of nodes in the network and try to select values that cover a wide range of realistic configurations. For details see `mixer_discovery.m`.

1.4.2 Tuning p_T

The model described in section 1.2 is simplified in two aspects: (1) Real-world capture behavior is more complex. (2) The assumptions on nodes' transmit behavior do not cover "the full story". Regarding the latter we consider two aspects:

- Coordinated transmit decisions: Modelling all details in this direction seems impossible. Our approach is: We assume that the relevant mechanisms in general improve the communication patterns including discovery. Hence, we regard the modelled results as bounds (somewhat).
- Wake-up: The simplified model does not consider that nodes must wake-up bevor they start to communicate. Therefore we tune p_T to support fast wake-up.

Our approach to tune p_T for fast wake-up is simple: In the first 4 slots after wake-up a node lower bounds its transmit probability by using $\min(p_T, 1 - s/4)$ instead of p_T where s is the number of slots passed since wake-up.

1.4.3 Leaving Discovery Phase

Mixer nodes use (2) to decide when to leave the discovery phase. This seems reasonable if p_D becomes sufficiently small. See `mixer_discovery.m` for details.

Note that this behavior causes another model mismatch because it means that beginning from a certail point in time not all nodes use p_T anymore (and nodes activate different coordination mechanisms). Again, we assume that this behavior does not disturb so much in the typical case.