Online Pairing of VoIP Conversations

Michail Vlachos [†] Aris Anagnostopoulos [‡] Olivier Verscheure [†] Philip S. Yu [†]

[†] IBM T.J. Watson Research Center
[‡] Yahoo! Research

Abstract

This paper answers the following question; given a multiplicity of evolving 1-way conversations, can a machine or an algorithm discern the conversational pairs in an online fashion, without understanding the content of the communications? Our analysis indicates that this is possible, and can be achieved just by exploiting the temporal dynamics inherent in a conversation. We also show that our findings are applicable for anonymous and encrypted conversations over VoIP networks. We achieve this by exploiting the aperiodic inter-departure time of VoIP packets, hence trivializing each VoIP stream into a binary time-series, indicating the voice activity of each stream. We propose effective techniques that progressively pair conversing parties with high accuracy and in a limited amount of time. Our findings are verified empirically on a dataset consisting of 1000 conversations. We obtain very high pairing accuracy that reaches 97% after 5 minutes of voice conversations. Using a modeling approach we also demonstrate analytically that our result can be extended over an unlimited number of conversations.

Keywords: stream clustering, binary time-series clustering, voice-over-ip, conversation pairing

1. INTRODUCTION

The International Telecommunications industry is in the early stages of a migration to Voice over Internet Protocol (VoIP). VoIP is a technology that enables the routing of voice conversations over any IP-based networks such as the public Internet. The voice data flows over a general-purpose packet-switched network rather than over the traditional circuit-switched Public Switched Telephone Network (PSTN). Market research firms including In-Stat and IDC predict that until 2009 we will experience the consumer and small business VoIP ramp-up period, and migration to VoIP will peak in the 2010-2014 time frame. According to recent reports, VoIP service revenue jumped jumped 66% to \$15.8 billion in 2006 after more than doubling in 2005, and is expected to more than triple by 2010 [1].

While the migration to VoIP seems inevitable, there are security risks associated with this technology that are carefully being addressed. Eavesdropping is one of the most common threats in a VoIP environment. Unauthorized interception of audio streams and decoding of signaling messages can enable the eavesdropper to tap audio streams in an unsecured VoIP environment. Call authentication and encryption mechanisms [5, 58] are being deployed to preserve customers' confidentiality. Preserving customers' anonymity is also crucial, which encompasses both the identity of the people involved in a conversation and the relationship caller/callee (pair of voice streams). Anonymizing overlay networks such as Onion Routing [22] and FindNot.com [59] aim at providing an answer to this problem by concealing the IP addresses of the conversing parties. A recent work [52] shows that tracking anonymous peer-to-peer VoIP calls on the Internet is actually feasible. The key idea consists in embedding a unique watermark into the encrypted VoIP flows of interest by minimally modifying the departure time of selected packets. This technique transparently compromises the identity of the conversing parties. However, the authors rely on the strong assumption that one has access to the customer's communication device, so that the watermark can be inserted before the streams of interest reach the Internet.

This work studies the feasibility of revealing pairs of *anonymous* conversing parties (caller/callee pair of streams) by exploiting the vulnerabilities inherent in VoIP systems. Our analysis also indicates that the proposed techniques are applicable even when the voice packets are encrypted. While the focus of this work is on VoIP data, the techniques presented here are of independent interest and can be used for pairing/clustering any type of binary streaming data. The contributions of the paper function on different levels:

1. We formulate the problem of pairing anonymous and encrypted VoIP calls.

2. We present an elegant and fast solution for the conversation-pairing problem, which exploits well established notions of complementary speech patterns in conversational dynamics.

Submitted to the VLDB Journal.

[1]`http://www.infonetics.com/resources/purple.shtml?ms07.vip.nr.shtml`

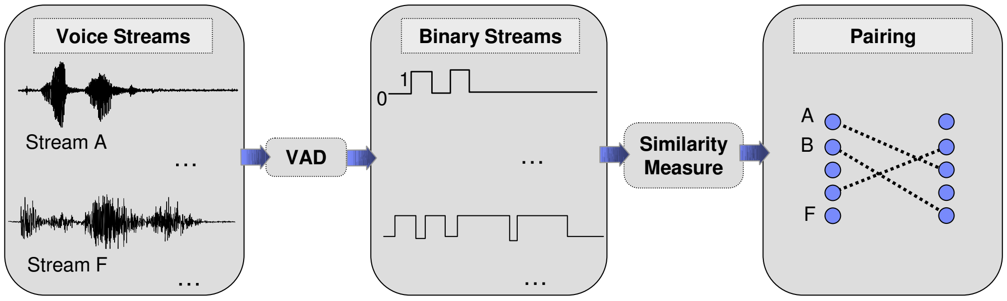

Figure 1: Overview of the proposed methodology

3. Our solution is based on an efficient transformation of the voice streams into binary sequences, followed by a progressive clustering procedure.

4. We provide a thorough analysis regarding the convergence and scalability of the presented algorithm.

5. Finally, we verify the accuracy of the proposed solution on a large dataset of voice conversations.

The paper is organized as follows [2]. Sections 2–5 present our solution for pairing conversations over any medium by mapping the problem into a complementary clustering problem for binary streaming data. We introduce various intuitive metrics to gauge the correlation between two binary voice streams and we present effective methods for progressively pairing conversing parties with high accuracy within a limited amount of time. Section 6 shows how the presented solution can be adapted for a VoIP framework, and demonstrates that encryption schemes do not hinder the applicability of our approach. In Section 7 we analyze through a modeling approach the convergence properties of the pairing algorithm. We validate the accuracy of the presented algorithm in Section 8 utilizing a large standard corpus of conversational speech data [23]. In Section 9 we present a proof of concept implementation of our approach within a real network monitoring system which is deployed on top of a parallel grid of computers. Potential solutions on the VoIP vulnerabilities that we have revealed is the topic of Section 10. Other applications of our pairing algorithms and similarity measures are are discussed in Section 11. Finally, we provide our concluding remarks in Section 12 and we also instigate directions for future work.

2. PROBLEM OVERVIEW AND METHODOLOGY

We start with a generic description of the conversation pairing problem, with the intention of highlighting the key insights governing our solution. We will later clarify the required changes so that the following model can be adapted for a VoIP environment.

2.1 Pairing Voice Conversations

Let us assume that we are monitoring a set $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of k voice streams (for now suppose k is even), comprising a total of $k/2$ conversations[3]. Each stream holds just one side of a two-way voice conversation, and there exists also a homologue voice stream that holds the other side of the conversation. Our objective is to efficiently reveal the relationship of *two-way* conversing parties. For example, assume S_2 is actually involved in a conversation with S_5. We aim at finding all relationships $S_i \leftrightarrow S_j$ including the example $S_2 \leftrightarrow S_5$, such that streams S_i and S_j correspond to each one-way voice stream of the same conversation. Our approach does not require an even number of voice streams and we also do not assume each voice stream to have a matching pair. Any voice stream without a corresponding counterpart is referenced to henceforth as a *singleton* stream. At the end of the pairing process some streams may remain unmatched. These will be the voice streams for which the algorithm either does not have adequate data to identify the conversational pair with high confidence, or simply streams whose pair does not exist in the pool of examined voice streams.

The key intuition behind our approach is that conversing parties tend to follow a "complementary" speech pattern. When one speaks, the other listens. This "turn-taking" of conversation [42] represents a basic rule of communication, well-established in the fields of psycholinguistics, discourse analysis and conversation analysis, and it also manifests under the term of speech "coordination" [13]. Needless to say, one does not expect a conversation to follow strictly the aforementioned rule. A conversational speech may well include portions where both parties speak or both are silent. Such situations are indeed expected, but in practice they do not significantly distort the results, since given conversations of adequate length, coordinated speech patterns are bound to dominate. We will show this more explicitly in the experimental section, where the robustness of the proposed measures are tested also under conditions of network latency.

Using the above intuitions, we will follow the subsequent steps for recognizing pairs of conversations:

1. First, voice streams are converted into binary streams, indicating the presence of voice (1) and silence (0).
2. Second, we leverage the power of complementary similarity measures, for quantifying the degree of coordination between pairs of streams.
3. Using the derived complementary similarity, we will employ a progressive clustering approach for deducing conversational pairs.

[2]This work represents an extension of article [51]

[3]In this paper we will not deal with multi-way conversations.

A schematic of the above steps is given in Figure 1. In the upcoming section we place our methodology within the context of the related work.

3. RELATED WORK

Recent work that studies certain VoIP vulnerabilities and has attracted a lot of media attention has appeared in [52]. The authors present techniques for watermarking VoIP traffic, with the purpose of tracking the marked VoIP packets. For accomplishing that, however, initial access to a user's device or computer is required. In this work we achieve a different goal; that of identifying conversational pairs, however we do not assume any access to a user's device. The only requirement of our approach (more explanations will be provided later) is the provision of a limited number of network *sniffers*, which will capture the incoming (encrypted) VoIP traffic.

Similar in spirit to our work is also [44], where the authors attempt to recognize which streaming movie/video a client is watching. They achieve that by extracting a bitrate signature of VBR encoded movies. Therefore, even though the content (movie) is encrypted the bitrate of the packets is still easily discernible, and can be compared against an already created database of bitrate signatures. The authors report that given a 40 minute trace, they are able to predict the movie correctly with over 77% accuracy, in a databases of 26 movies.

There is also extended related work on clustering of data streams. Guha, et al. [24] have studied the k-Median problem utilizing a one-pass algorithm and small space. Other approaches on the same topic have been presented by Babcock, et al. in [3] and by Charikar, et al. in [12]. Methodologies for clustering data streams have also been presented in [37]. The above streaming algorithms sacrifice accuracy in favor of efficiency in execution and storage space. [17] examines general ways of scaling up machine learning techniques, and in specific how to scale up the K-Means algorithm utilizing Hoeffding bounds for determining the necessary sample size. CluStream [1] considers stream clustering at different temporal granularities, while HPStream [2] examines clustering of high-dimensional streams and focuses only on the most meaningful dimensions. Clustering approaches that deal specifically with binary streams have also appeared in [38, 34].

The objective of traditional streaming clustering algorithms is on the maintenance of the cluster centers, while in this work once a cluster (pair of streams) is discovered it is removed from further examination. Therefore, while the previous algorithms achieve speedup by sacrificing accuracy and retaining the necessary statistics in reduced storage space, our speedup is attributed to the incremental (online) nature of algorithm, while also leveraging the removal of paired streams for additional performance boost. Finally, the majority of the above algorithms perform the clustering within a single stream, while our focus is on clustering across multiple streams.

Algorithms that operate on multiple streams have appeared in [57], [39]. StatStream [57] focuses on the online monitoring of the most correlated pair of streams, primarily for financial (stock-market) applications. SPIRIT [39] presents techniques for incremental computation of the SVD transform between multiple streams, but it does not consider the computation of correlation or anti-correlation especially for all pairs of streams.

The topic of efficient similarity execution between streams, has also been of interest in the research community; in [15], Cormode et al., study the use of similarity measures for comparison of streams and focus on sketch approximations of the Hamming Norm. Various other norm approximations, such as dominance norms [16] and entropy norms [11], have also appeared in the related bibliography. This work examines the use of *complementary* similarity measures between binary streaming data, which can also be seen as discovery of the most anti-correlated pairs between multiple streams. Therefore, related to the current problem is also the work of Sakurai, et. al [43], that considers the problem of streaming discovery of lag correlations (which is however within a single stream).

The online clustering algorithm presented in this work has several unique characteristics, such as: 1) Consideration of clusters as *pairs* of objects (streams), which is one of the core requirements for the application that we examine. 2) Anytime nature of the algorithm, returning progressively identified pairs of streams before the complete execution of the algorithm. 3) Gradual removal of paired streams, which also results in the enhanced algorithm performance.

The methods presented in this work, exploit and adapt data-mining techniques for depicting inherent vulnerabilities in VoIP streams, which can potentially compromise the users' anonymity. It is interesting to note, that much of recent work in data-mining [31, 14] has focused on how to embed or maintain privacy for various data-mining techniques, such as clustering, classification, and so on. Recently, Sion, et al., have also presented techniques for authenticating streams of data [46].

In the sections that follow we will provide a concise description of a Voice Activity Detector. We will also present intuitive *coordination* measures for quantifying the complementary similarity between binary streams. We put forward a lightweight pairing technique based on adaptive soft decisions for reducing the pairing errors and avoiding the pairing of singleton streams. Key requirements include lightweight processing, quick and accurate identification of the relationships, and resilience to both latency and noise.

4. MEASURES OF SPEECH COORDINATION

Our goal is to identify coordinated speech patterns that indicate the presence of a two-way conversation. First we examine how to convert a voice stream into a binary stream indicating the portions of speech and silence, and later we present various complementary similarity measures for the resulting binary streams.

4.1 Voice Activity Detection

The goal of a Voice Activity Detection (VAD) algorithm [40] is to discriminate between voiced versus unvoiced sections of a speech stream. We provide only a high-level description of a typical VAD algorithm for reasons of completeness, since it is not the focus of the current work. The VAD process computes the energy of small overlapping speech packets (also called *frames*, with each frame being 20-30msec in length), and employs an adaptive energy threshold that will differentiate the voiced from the unvoiced frames. The threshold is typically deduced by estimating the average energy of the unvoiced portions, taking also into consideration a background noise model, based on the characteristics of the data channel. The output of the VAD algorithm will be "1" when there is speech detected and "0" in the presence of silence. A simple schematic of its operation is provided in Fig. 2.

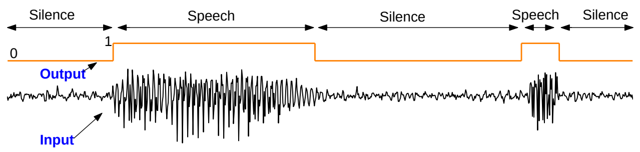

Figure 2: A Voice Activity Detector can effectively recognize the portions of silence or speech on a voice stream.

As will be explained later, the voice activity detection is inherently provided by the VoIP protocol.

4.2 Complementary Similarity

After voice activity detection is performed, each voice stream S_i is converted into a binary stream B_i. The resulting binary stream only holds the necessary information that indicates the speech/no-speech patterns. The objective now is to quantify the *complementary similarity* (which we call *cimilarity*) between two binary streams.

As already mentioned, the basic insight behind detecting conversational pairs is to discern voice streams that exhibit complementary speech behavior. That is, given a large number of binary streams $B_1, B_2, \ldots, B_N$, and a query stream B_q (which indicates the voice activity of user q), we would like to identify the stream B_j that is most complementary similar to stream B_q, or in other words, has the largest cimilarity.

We present different versions of cimilarity measures (*Cim*) and we quantify their performance later on, in the experimental section. Let us consider two binary streams, B_i and B_j. By abstracting B_i and B_j as binary sets, an intuitive measure of coordination between users i and j consists in computing the intersection between B_i and the binary complement of B_j normalized by their union. We denote by *Cim-asym*(i, j, T) this measure computed over streams B_i and B_j after T units of time. One can readily verify that it can be written as:

$$\text{Cim-asym}(i, j, T) = \frac{\sum_{t=1}^{T} B_i[t] \wedge \neg B_j[t]}{\sum_{t=1}^{T} B_i[t] \vee \neg B_j[t]}. \qquad (1)$$

where $B_k(t) \in \{0, 1\}$ is the binary value for user k at time t, and the symbols $\wedge$, $\vee$, and $\neg$ denote the binary AND, OR and NOT operators, respectively.

Note that Equation (1) asymmetrically measures the amount of coordination between speakers i and j. That is, in general, *Cim-asym*$(i, j, T) \neq$ *Cim-asym*(j, i, T) due to the complement operator on binary data. This measure can be seen as the asymmetric extension of the well-known Jaccard coefficient [10]. Thus, we also refer to this measure as *Jaccard-Asymmetric*.

$B_i \setminus B_j$	1	0
1	0	1
0	0	0

$B_i \setminus B_j$	1	0
1	1	1
0	0	1

Figure 3: Computation of `Cim-asym`, *Left:* Numerator, *Right:* Denominator

Computing `Cim-asym` between two binary streams is computationally very light. The computation lookup table for the numerator and the denominator is provided in Figure 3. The numerator is increased when $B_i = 1$ and $B_j = 0$, while the denominator is not increased when $B_i = 0$ and $B_j = 1$. So *Cim-asym*(B_i, B_j) only rewards the presence of non-speech of user j, when user i speaks.

Example: Given $B_1 = 11100110$ and $B_2 = 00010001$ then *cim-asym*(B_1, B_2) = 0.833 and *cim-asym*(B_2, B_1) = 0.6667.

The `Cim-asym` measure is also easily amenable to incremental maintenance as time T progresses. Indeed, let $V_\wedge(i, j)$ and $V_\vee(i, j)$ denote the running values of the numerator and the denominator, respectively. The value of *Cim-asym*(i, j, T) for any elapsed time T is given by the ratio $V_\wedge(i, j)/V_\vee(i, j)$. Therefore, given n binary streams, incrementally computing *Cim-asym* requires keeping 2 times $n(n-1)$ values in memory. For example, when monitoring $n = 1000$ streams and assuming each value is stored as an *16 bit integer*, only 4 MBytes of memory are needed for tracking all the required statistics.

We also consider a symmetric extension of *Cim-asym*. This metric is even simpler than the previous one and is essentially a (scaled) hamming distance; we denote it as *Cim-Ham* and we refer to it as *Hamming-Scaled*:

$$\begin{aligned}\text{Cim-Ham}(i, j, T) &= \sum_{t=1}^{T} \frac{(B_i[t] \wedge \neg B_j[t]) \vee (\neg B_i[t] \wedge B_j[t])}{T} \\ &= \sum_{t=1}^{T} \frac{\text{XOR}(B_i[t], B_j[t])}{T},\end{aligned} \qquad (2)$$

Given n binary streams, incrementally computing *Cim-Ham* requires keeping only $\binom{n}{2}$ values in memory due to its symmetric nature. Using the example above, memory requirements drop to approximately 1 Mbytes given the same assumptions. The *Cim-Ham* is generally more aggressive than the asymmetric Jaccard, because it also rewards the presence of speech patterns when the user in question does not speak. However, our experiments indicate that the most conservative asymmetric Jaccard metric ultimately achieves the best detection accuracy.

Finally, we consider the *Mutual Information* (MI) as a measure of coordination between conversing parties [4]. Approximations of the entropy between streams have also recently appeared in [11]. Entropy is a measure indicating how much information can be obtained about one random variable B_i by observing another B_j. Let $p_{i,j}(x,y)$, $p_i(x)$, and $p_j(y)$ with $x,y \in 0,1$ denote the joint and marginal running averages for users i and j after T units of time. For example,

$$p_{i,j}(0,1) = \frac{1}{T}\sum_{t=1}^{T} \neg B_i[t] \wedge B_j[t].$$

The amount of *Mutual Information* (MI) between streams B_i and B_j is written as:

$$\text{MI} = \sum_{x,y \in 0,1} p_{i,j}(x,y) \log_2 \frac{p_{i,j}(x,y)}{p_i(x)p_j(y)} \qquad (3)$$

The mutual information measure requires higher processing power but exhibits symmetry. Note that while at first it seems that one needs to store 8 statistics for updating the Mutual Information, in fact only 3 statistics are required. For example $p_{i,j}(0,0)$, $p_{i,j}(0,1)$ and $p_{i,j}(1,0)$ are sufficient to restore the remaining ones, since:

$$\begin{aligned} p_{i,j}(1,1) &= 1 - p_{i,j}(0,0) - p_{i,j}(0,1) - p_{i,j}(1,0) \\ p_i(0) &= p_{i,j}(0,0) + p_{i,j}(0,1) \\ p_j(0) &= p_{i,j}(0,0) + p_{i,j}(1,0) \end{aligned}$$

and so on.

So, given n binary streams, it can be shown that incrementally computing *MI* requires keeping 3 times $\binom{n}{2}$ values in memory thanks to its symmetric nature. Thus, approximately 3 MBytes of memory are required for the above example, when monitoring 1000 streams.

In the following section, we illustrate how any of the above metrics can be used in conjunction with a progressive clustering algorithm for identifying conversing pairs.

5. CONVERSATION PAIRING/CLUSTERING

In order to get insights about the pairing algorithm we first plot how the complementary similarity of one voice stream progresses over time against all other streams. In Figure 4 we depict the progression of cimilarity of voice streams A and B against all other streams, using a dataset containing 280 data streams.

One can notice that voice pairing is extremely ambivalent during the initial stages of a conversation, but the uncertainty decreases as conversations progress. This is observed, first, because most conversations in the beginning exhibit a customary dialog pattern ("hi," "how are you," etc.). However, conversations are bound to evolve in different conversational patterns, leading to a progressive decay in the matching ambiguity. Second, some time is required to elapse, so that the probability that two unrelated streams match by chance becomes sufficiently low.

Intuitively, better discrimination between conversations is provided as additional voice data are collected. Therefore, a simple but effective solution for tackling the conversation pairing problem would be to compute the pairwise cimilarity matrix M after some time T, where each entry provides the complementary similarity between two streams:

$$M(i,j) = \text{Cim}(i,j,T),$$

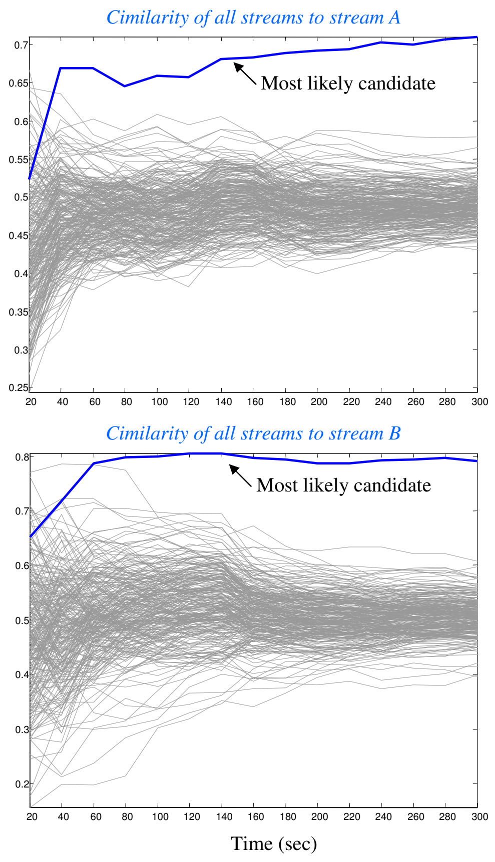

Figure 4: Progression of complementary similarity (*Jaccard-Asymmetric*) over time for voice sequences A and B, against all other communications. The most likely pairing sequence can be visualized as the one with the highest cimilarity.

where Cim is one of the cimilarity measures that we presented in Section 4.

Then we can pair users i and j if we have

$$M(i,j) = \max_{\ell}\{M(i,\ell)\}$$

and

$$M(i,j) = \max_{\ell}\{M(\ell,j)\}.$$

We call this approach *hard* clustering, because at each time instance it provides a rigid assignment of pairs, without providing any hints about the confidence or ambiguity of the matching. A pseudocode for this approach is provided in Figure 6. This rigid clustering approach finds pairs for all streams. Therefore, situations like the ones in Figure 5 will also be assigned, even though there is no clear separation.

Additional shortcomings can also be identified with the above hard clustering approach:

- First, it provides no concrete indication when the pairing should start. When are the sufficient statistics robust enough to indicate that pairing should commence?
- In order to achieve high accuracy, sufficient data need to be collected. This penalizes the system responsiveness (no decision is made until then) and additionally significant resources are wasted (memory and CPU).

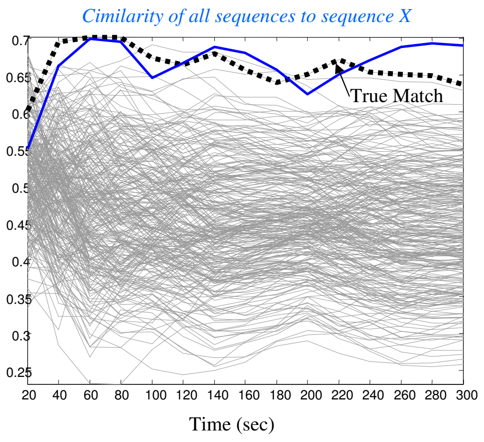

Figure 5: A case where no conclusion can be safely deduced. The sequence with the maximum cimilarity after 300sec is not the true pair of sequence X.

- Different streams will converge at different rates to their expected similarity value. Therefore, decisions for different pairs of streams can (or cannot) be made at different times, which is not exploited by the hard clustering approach.

```
1.  **Function** matchStreams($\mathcal{S}$)
2.      /* $\mathcal{S}$ contains all the streams $[1:N]$ */
3.      compute the pairwise cimilarities $M(s_i, \cdot)$
4.      $\max_1 \leftarrow \max_\ell M(s_i, \ell)$
5.      $sm_i \leftarrow$ sequence ID of $\max_1$
6.      match sequence $s_i$ matches with sequence $sm_i$
```

Figure 6: Hard matching of conversations

For a dataset with n voice sequences, each of length m, the time complexity of this rudimentary pairing algorithm is $O(n^2m)$. In the pairing algorithm that we describe below, we will address all the previous issues, allowing the early pairing of streams, while imposing minimal impact on the system resources.

Using as a guide the aforementioned behavior which governs the progression of cimilarity, we construct the clustering algorithm as an *outlier detection scheme*. What we have to examine is whether the closest match is "sufficiently distant" from the majority of streams. Therefore, when comparing a stream against all others, the most likely matching candidate should not only hold the maximum cimilarity, but also deviate sufficiently from the cimilarity of the remaining streams.

Figure 7 contains a pseudocode of the pairing algorithm, while Figure 9 depicts the steps behind its execution. We maintain the same matrix M as in the hard-clustering approach, which is updated as time progresses. Then, at every step of the algorithm, for every stream that has not been matched, we perform the following actions. Suppose that at any time T we start with the binary stream B_1:

1. We perform a k-trim by removing the k most distant and k closest matches (typically $k = 2, \ldots, 5$).
2. We compute the average cMass (center of mass) of the remaining stream cimilarities.
3. We record the cimilarity of the two closest matches to stream B_1, which we denote as $\max_1$ and $\max_2$. We consider the closest match "sufficiently separated" from the remaining streams if the following holds:
$$\max_1 - \max_2 > f \cdot (\max_2 - \text{cMass}),$$
where the f constant captures the assurance (confidence) about the quality of our match. Typical values of f range within $[0.5, 2]$. Greater values of f, signify a more separable best match compared to the remaining streams, and hence more confident pairing of streams.
4. If the above criterion does not hold we cannot make a decision about stream B_1. Otherwise we match B_1 with $\max_1$ and we remove their corresponding rows and columns from the pairwise cimilarity matrix (Figure 8).

```
1.  **Function** matchStreams($\mathcal{S}$, $f$)
2.      /* $\mathcal{S}$ contains all the streams $[1:N]$ */
3.      **for** ($t = 0, 1, 2, \ldots, T$)
4.      /* $T$ is an upper bound that will depend on $n$ (Ideally,
         $T = \Theta(\ln n)$) */
5.          Update the pairwise similarity matrix $M(\cdot, \cdot)$
6.          **foreach** unmatched stream $s_i$
7.              compute $\max_1$, $\max_2$ of $M(s_i, \cdot)$
8.              $sm_i \leftarrow$ stream ID of $\max_1$
9.              **trimmedMsi** $\leftarrow$ $k$-trim of $M(s_i, \cdot)$
10.             cMass $\leftarrow$ mean of **trimmedMsi**
11.             **if** ($\max_1 - \max_2 > f \cdot (\max_2 -$ cMass$)$)
12.             /* stream $s_i$ matches with stream $sm_i$ */
13.                 remove rows $s_i$, $sm_i$ from $M$
14.                 remove columns $s_i$, $sm_i$ from $M$
15.         **if** (all streams are paired) **return**
```

Figure 7: Progressive algorithm for matching streams.

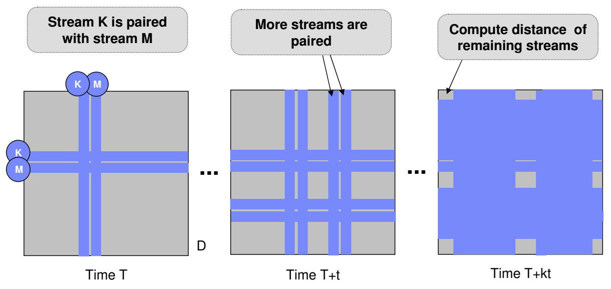
Stream K is paired with stream M
More streams are paired
Compute distance of remaining streams
K
M
K
M
D
...
...
Time T
Time T+t
Time T+kt

Figure 8: The similarity/dissimilarity matrix does not have to be completed fully, for all time instances. Matching pairs can be removed from computation.

Notice that the *outlier detection criterion* adapts according to the current similarity distribution, being more strict in the initial phases (wider $(\max_2 - \text{cMass})$) and becoming more flexible as time passes.

Furthermore, the algorithm does not need to know a priori a bound on the required time steps for execution. As soon as there is sufficient information, it makes use of it and it identifies the candidate pairs.

Next, we analyze the performance of the algorithm.

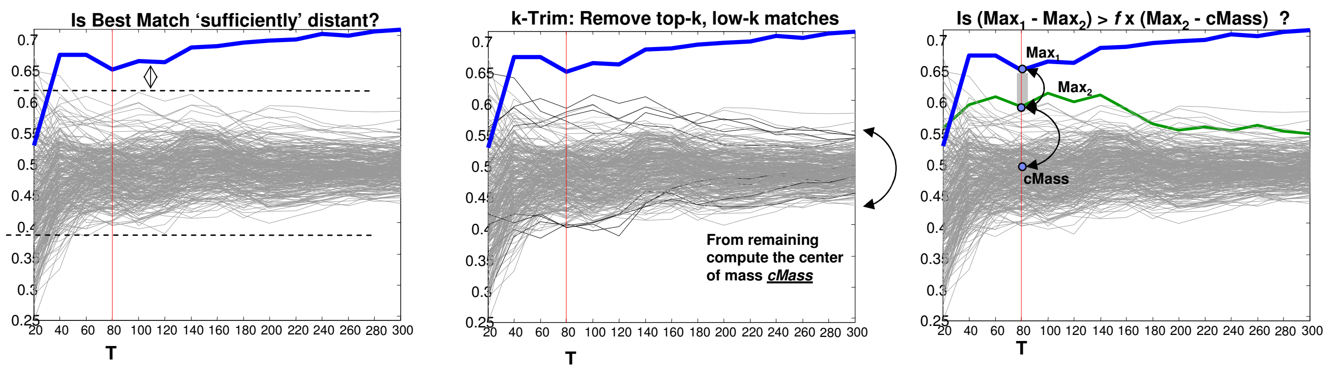

Figure 9: Pairing of voice conversations.

5.1 Time and Space Complexity

Compared to the hard-clustering approach that needs to recompute the pairwise similarity matrix M for each time step, the progressive algorithm reduces the computational cost by progressively removing from the distance computation the streams that have already been paired, although the initial stages of the algorithm are somewhat more expensive, since in every iteration there are more operations performed than just the update of the similarity matrix M.

So, let us analyze the time and space that our algorithm requires. We assume that we execute the algorithm with an initial set of n streams, so the size of matrix M is n^2, hence the space requirement is $O(n^2)$.

For the time complexity, assume that at time step t there are S_t streams available. Then, the running time required to execute the t^{th} step is $O(S_t^2)$. To see that, notice that line 5 of Figure 7, where we describe the update of cimilarity matrix M, requires $O(S_t^2)$ time steps. The **foreach** loop at line 6, where we process each stream, is executed at most S_t times and each of the commands inside the loop can be computed in linear (in S_t) time. (The most involved is line 9 for computing the k-trim, which can be done with a variation of a linear algorithm for computing the median.) Therefore, the running time of every time step of the algorithm is $O(S_t^2)$, in other words there exists a constant κ such that the time per step is bounded by $\kappa \cdot S_t^2$.

Therefore, if T_r is the total running time, we have

$$T_r \leq \kappa \cdot \sum_{t=1}^{\infty} S_t^2.$$

It is now clear that the running time depends on the rate with which streams become paired. If the streams become easily distinguishable at early stages, then they are removed and S_t decreases fast; otherwise the running time can be arbitrarily large. However, in practice, convergence is fast as we also indicate with several experiments in Section 8. Figure 10 depicts visually the pairing process and the lifecycle of various streams for an experiment with 280 voice streams. We plot the cimilarity of one voice stream against all others and each line represents a voice stream and is extended only up to the point when the stream is paired. The true match for the examined stream, is indicated by the thick line, and is discovered after 240sec, when only 4 streams remain unpaired.

In Section 7 we also present a model for conversation patterns, where we show that in $O(\ln n)$ time steps with high probability our cimilarity measures can distinguish all pairs. Since $S_t \leq n$, we can conclude that given that model, the total number of time steps is bounded by $O(n^2 \ln n)$, with high probability. Therefore, the algorithm is efficient, because it only introduces an $O(n \log n)$ complexity per data stream.

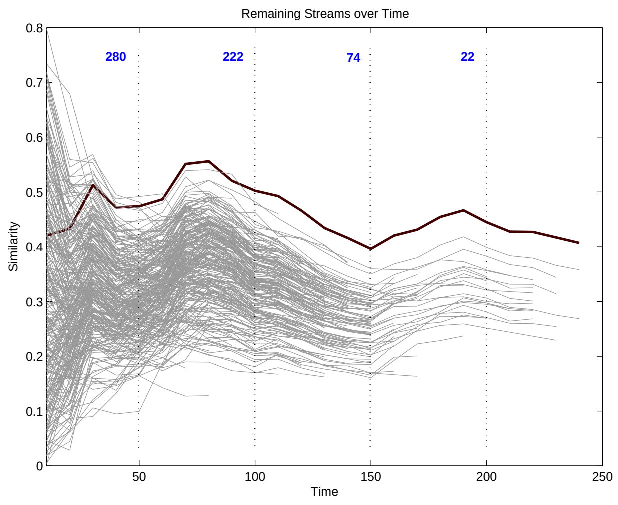

Figure 10: We show the number of remaining streams at each time instance on an experiment with 280 streams. The darker stream indicates the actual best match which is identified after 240 seconds.

6. EXTENDING TO A VOIP NETWORK

We explain how the previous model of pairing voice conversations can be extended to work on a voice-over-IP network. In what follows we describe the structure and transmission protocol of a typical VoIP network and we illustrate the steps for reconstructing the binary voice activity stream from a sequence of VoIP packets.

We consider the framework depicted in Figure 11. N VoIP subscribers are connected to the Internet either directly via their ISP providers, or behind VoIP gateways on traditional PSTN networks. Those VoIP subscribers may use a low-latency anonymizing service composed of a set of overlay network nodes. Each VoIP stream traverses a possibly distinct set of IP routers, a subset of which are assumed to have VoIP sniffing capabilities. Each sniffer pre-processes the incoming VoIP traffic and forwards the resulting data to a central processing unit.

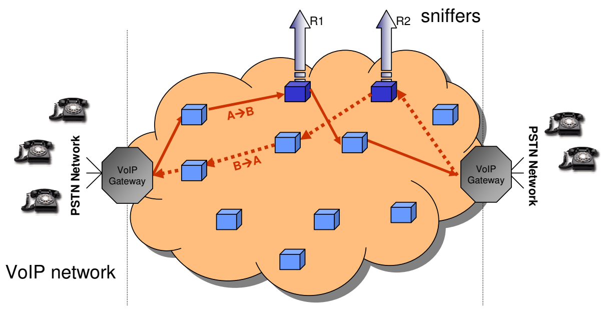

Figure 11: VoIP Framework. A set of customers with direct access to the IP network or behind a PSTN network. A set of IP routers (light boxes), a subset of which are VoIP sniffers (dark boxes) that forward preprocessed VoIP data to a Central Processing Unit.

A voice signal captured by a communication device goes through a series of steps in preparation for streaming. Figure 12 summarizes some of the following concepts. A voice signal is continually captured by the microphone of a communication device. The digital signal is segmented and passed over to a Voice Activity Detection (VAD) unit. This feature allows VoIP devices to detect whether the user is currently speaking or not by analyzing voice activity. Whenever the voice activity is below a certain adaptive threshold, the current segment is dropped. Note that if the VAD algorithm is not sophisticated enough, actual voiced segments may get wrongly filtered out [41]. The filtered signal is then passed through a voice codec unit (e.g., G.729.1 or GSM) that compresses the input voice segments to an average bitrate of approximately 10 Kbps. Those compressed segments are encrypted using 256-bit AES [5] and packetized using the Real-time Transport Protocol (RTP). Each RTP packet consists of a 12-byte header followed by 20ms worth of encrypted and compressed voice. It is important to note that all RTP headers are in the clear [5]. Various RTP header fields are of great interest for our purpose. In particular, the Payload Type (PT) field enables easy spotting of VoIP streams, the Synchronization Source (SSRC) field uniquely identifies the stream of packets, and the Timestamp field reflects the sampling instant of the first byte in the RTP payload. Finally, each RTP packet is written to a network socket.

6.1 Separating the Voice Streams

For recasting the problem into the setting that we previously studied, we need first to reconstruct the binary streams indicating the voice activity of each one-way communication. Given the above transmission protocol, a VoIP sniffer that gathers incoming internet traffic can identify and separate the different voice streams and also convert them into binary streams that indicate periods of activity or silence as follows:

1. The RTP PT field is used to segregate VoIP packets from different data traffic (see Figure 13).

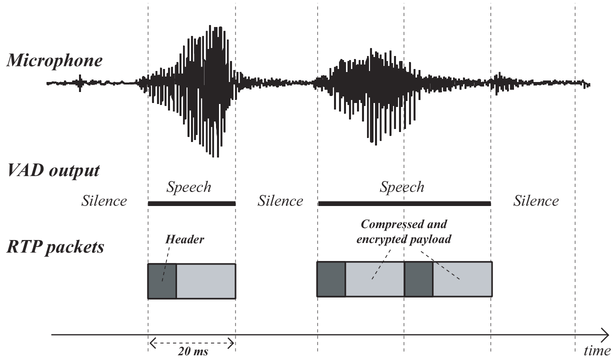

Figure 12: A voice signal captured by a communication device goes through various steps in preparation for streaming.

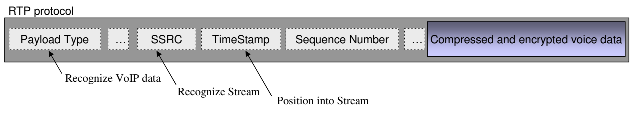

Figure 13: Fields of the RTP protocol that are used

2. Each different voice stream can be tracked by its unique RTP SSRC field.

3. Finally, the binary stream indicating the presence of speech or silence, is inherently provided by the VoIP protocol, given the presence or not of a voice packet. Packets are only sent during speech activity which constitutes an indirect way of reconstructing the binary voice activity stream. For a given RTP SSRC value (stream ID), a sniffer measures the difference of two consecutive Timestamp values (inter-departure time of packets) and generates a one (or a zero) if the difference is equal to (or larger than) the segmentation interval of 20ms. Thus, each binary stream results from the aperiodic inter-departure time of VoIP packets derived from the Voice Activity Detection (VAD), which is performed within the customer's communication device. In Figure 14 we visualize this process.

6.2 Advantages and Discussions

Several are the advantages of the aforementioned methodology:

- A very important first outcome, is that the technique operates on the compressed data domain. Because we do not need to decompress the voice data to perform any action, this immediately gives a significant performance advantage to the approach.

- A second observation, is that the presented methodology is also valid even when the voice data is encrypted. This is true because for performing voice activity detection we merely exploit the presence of the packet as an indication of speech. Note, that because the data can still be encrypted, the privacy of the conversation content is not violated.

- Finally, the presented algorithm is robust to jitter and network latency. Jitter has no effect on the RTP Time-

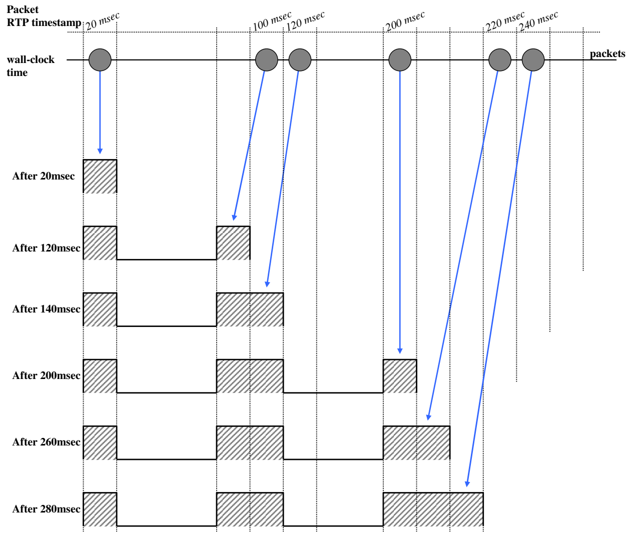

Figure 14: Usage of RTP Timestamp for reconstructing the binary voice activity sequence

stamp values, which are assigned during the data transmission and measure the *inter-departure* packet time. Network latency only affects the arrival of the first packet, since synchronization of subsequent packets can be reconstructed by the corresponding RTP timestamps. In our experiments we do not assume a zero-latency network. Instead we show that our method is indeed resilient to latency.

We briefly elaborate on certain issues or questions that may arise given the dynamic nature of the system:

a) We do not assume that the network sniffers are able to track all voice streams. Singleton streams can be present. This does not pose a problem for our algorithm since we do not enforce pairing of all streams. We depict this fact lucidly in the experimental section.

b) The cardinality of voice conversations captured by the sniffer changes over time as calls start and/or terminate. Therefore, one should pair streams that commence at approximately the same time (within twice the assumed worse network latency). This gives rise to multiple cimilarity arrays formed by voice streams with similar arrival times. We demonstrate this notion in Figure 15. In the experiments we only consider the case where all k voice streams are concurrent, since this illustrates better the scalability and accuracy of our approach under the maximum possible load.

c) Finally, it is worth noting that the RTP Sequence Number field together with the Timestamp values help avoid blindly concluding a packet has been filtered out by the VAD unit while, in fact, it has been dropped by the network. However losses are seldom in commercial VoIP networks and in this work we do assume a lossless VoIP framework.

7. THEORETICAL ANALYSIS

We present a theoretical analysis on the performance of the pairing protocol. The objective is to demonstrate the fast (and accurate) convergence of the pairing algorithm. While in the experimental section we will demonstrate the pairing

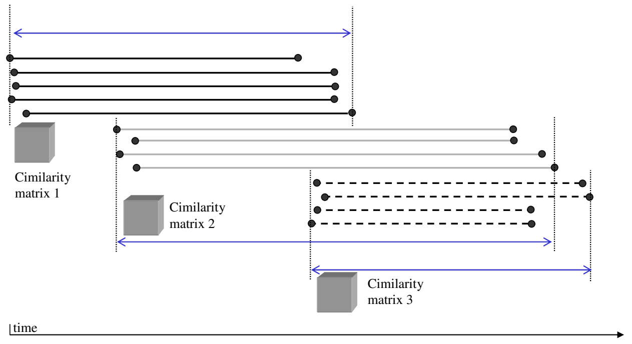

Figure 15: Asynchronous arrival of multiple voice streams at the sniffer. Creation of multiple cimilarity arrays for streams that are first captured at approximately the same time.

accuracy for 1000 real voice conversations, a natural question that arises is whether accuracy is compromised when the matching algorithm is challenged with an increasing number of voice streams. This section answers specifically this question. Our analysis indicates that the necessary time to correctly pair all the streams increases only *logarithmically* in the number of streams, therefore the accuracy of the pairing algorithm scales very gracefully with larger problem instances.

7.1 Model

In order to capture and model the relationship between the pairwise complementary similarity between various conversations, we need first to assume a model for 0/1 VAD patterns that are used as input for our clustering algorithm.

The simplest generative model for the 0/1 patterns, would be to assume a Bernoulli process, where $\mathbf{Pr}(B_i[t] = 1) = p_{ij}$ and $\mathbf{Pr}(B_i[t] = 0) = 1 - p_{ij}$, if users i and j are participating in a conversation. However, a plot of the distribution of 0's and 1's clearly suggests, that a more accurate model should follow an exponential trend (see Figure 16).

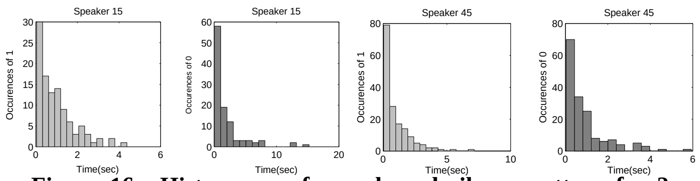

Figure 16: Histograms of speech and silence pattern for 2 speakers.

Our observations are coherent with the related bibliography, which utilizes exponential on-off sources for modeling voice traffic [9, 29, 36]. Therefore, for every two users participating in a conversation, we assume that they alternate between conversation segments (a segment where one of the two user transmits and the other receives is followed by a segment that the first user receives and the second one transmits, and so on), and the duration of the segment that user i transmits follows a geometric distribution (the discrete analog of the exponential distribution) with parameter p_i.

This generative model is equivalent to a process that is controlled by the two-state Markov chain of Figure 17. At state “SR” user i sends (speaks) while his/her partner receives (listens) and at state "RS" user i receives and the other party sends. Similar approaches have also be employed for traffic modeling over telephone networks [9, 29, 36]. One can conceive more complex models where there are states where no user speaks or where both users speak concurrently; the techniques of this section can extend to those as well.

The aforementioned Markov chain has a stationary distribution, and the probability under the stationary distribution that user i is transmitting equals

$$r_i = \frac{1 - q_i}{2 - p_i - q_i}, \tag{4}$$

where we define $q_i = p_j$ if user i is involved in a conversation with user j; the reader can verify that Equation (4) satisfies the equation

$$\begin{bmatrix} r_i \\ 1 - r_i \end{bmatrix} = \begin{bmatrix} p_i & 1 - p_i \\ 1 - q_i & q_i \end{bmatrix}^T \cdot \begin{bmatrix} r_i \\ 1 - r_i \end{bmatrix}.$$

Furthermore, in order to accommodate for errors in transmission or for two users talking or pausing simultaneously, we augment the model by adding some noise. In particular, we assume that every bit of every stream is flipped with some probability θ ($\theta < 1/2$), and that all the streams and all the time points are independent.

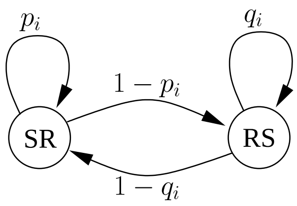

Figure 17: The two-state Markov chain that describes the talking alternating process between user i and the conversing party.

7.2 Computing the Cimilarity Measure

Now we prove our main result, Theorem 7.1, which shows that the cimilarity measures are able to correctly match the conversing users using only a small amount of collected data, and with low probability of error. After the presentation of the theorem and its proof, we comment on the theorem implications.

THEOREM 7.1. *Assume that there are n users participating in $n/2$ conversations according to the communication model of Section 7.1. Then in $O(\ln n)$ time steps a cimilarity measure can distinguish the correct pairings with high probability*[4].

PROOF. We present the analysis of *Jaccard-Asym*. We focus on stream 1 and we assume that we try to find the pairing communication stream. The model described in the previous section induces a Markov chain, and the state in which the chain is at time t contains information about the states of each pair, as well as about which bits have been flipped due to noise. Furthermore, the evolution of the streams of user 1 and user i—whether participating in a conversation or not—induces a Markov chain $\{X(t) : t = 1, 2, \dots\}$ on the state space $\Sigma = \{(S_1, S_i, E_1, E_i)\}$, with $S_1, S_i \in \{S, R\}$, $E_1, E_i \in \{N, Y\}$, where S_1 (S_i) indicates whether user 1 (i) transmits or receives and E_1 (E_i) indicates whether there is an error in the stream of user 1 (i) at the current time point (i.e., if the bit of the user is flipped). The transition probabilities of the Markov chain are defined by the probabilities that users 1 and i switch their state from S to R or vice-versa, and the probabilities of a bit being flipped due to noise. For example, if users 1 and i are not involved in a conversation we have

$$\begin{aligned}\mathbf{Pr}(X_{t+1} = (S, R, Y, N) \mid X_t = (S, S, N, Y)) = \\ = p_1(1 - p_i)\theta(1 - \theta)\end{aligned}$$

This Markov chain is irreducible and aperiodic, so it has a stationary distribution, say π. Then we can compute, for example,

$$\pi((S, R, N, Y)) = r_1(1 - \theta)\theta,$$

if users 1 and i are conversing and

$$\pi((S, R, N, Y)) = r_1(1 - r_i)(1 - \theta)\theta,$$

if they are not. Instead of computing them explicitly, one may see that these expressions follow from Equation (4), by interpreting the stationary probability of a state as the fraction of time that the chain spends in the state as time goes to infinity.

Furthermore, we define a function $B_i : \Sigma \mapsto \{0, 1\}$, which indicates the bit at the current time point of users 1 and i; for example, $B_1(S, S, N, Y) = 1$ and $B_i(S, S, N, Y) = 0$. Let $\Sigma_1 \subset \Sigma$ be the set of states σ for which $B_1(\sigma) = 1$ and $B_i(\sigma) = 0$ (i.e., $\Sigma_1 = \{(S, S, N, Y), (S, R, N, N), (R, S, Y, Y), (R, R, Y, N)\}$) and let $\Sigma_2 \subset \Sigma$ be the set of states σ for which $B_1(\sigma) = 1$ or $B_i(\sigma) = 0$. We can also compute the probability of Σ_1 and Σ_2, for example,

$$\begin{aligned}\pi(\Sigma_1) &= \pi((S, S, N, Y)) + \pi((S, R, N, N)) + \\ &\quad \pi((R, S, Y, Y)) + \pi((R, R, Y, N)).\end{aligned}$$

Notice then that the complementary similarity of streams 1 and i during the time interval $[1, T]$ is given by the expression

$$R_i(T) = \text{Cim-asym}(1, i, T) = \frac{\frac{1}{T}\sum_{t=1}^{T} \mathbf{1}_{\Sigma_1}(X_t)}{\frac{1}{T}\sum_{t=1}^{T} \mathbf{1}_{\Sigma_2}(X_t)},$$

where $\mathbf{1}_S(\cdot)$ is the indicator function of event S (i.e., $\mathbf{1}_S(x) = 1$ if $x \in S$ and 0 otherwise).

Since the Markov chain is ergodic (irreducible and aperiodic), the above summations converge to their expectation under the stationary distribution, for example,

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}_{\Sigma_1}(X_t) = \pi(\Sigma_1),$$

where π is the stationary distribution of the Markov chain. A similar result holds for the summation at the denominator. Therefore, in order to show that our algorithm succeeds in

[4]We say that a limiting statement holds "with high probability" (abbreviated "whp.") if it holds with probability that is at least $1 - 1/n^c$ for some constant $c > 0$.

distinguishing the user that communicates with user 1 from the rest of the users, we show first, that the values R_i differ by a significant amount depending on whether user i is involved in a communication with user 1 or not. Furthermore, we have to show that the convergence to these values takes place fast, and for that we will demonstrate that for a relatively small value of T the ratio $R_i(T)$ is close to the value under the stationary distribution for every user i.

By using Equation (5) (and the pertinent equation for Σ_2) we get after some tedious calculations:

$$\begin{aligned} \mathrm{E}_\pi[R_i] &= \mathrm{E}_\pi[\text{Cim-asym}(1,i)] \\ &= \frac{\pi(\Sigma_1)}{\pi(\Sigma_2)} = \frac{r_1 + \theta^2 - 2r_1\theta}{r_1 + 2\theta - 2r_1\theta - \theta^2} \end{aligned}$$

if user i is the party conversing with user 1, and

$$\mathrm{E}_\pi[R_i] = \mathrm{E}_\pi[\text{Cim-asym}(1,i)] = \frac{r_1 - r_1r_i + \theta - 3r_1\theta - r_i\theta + 4r_1r_i\theta - \theta^2 + 2r_1\theta^2 + 2r_i\theta^2 - 4r_1r_i\theta^2}{1 - r_i + r_1r_i - \theta + r_1\theta + 3r_i\theta - 4r_1r_i\theta + \theta_2 - 2r_1\theta^2 - 2r_i\theta^2 + 4r_1r_i\theta^2},$$

if user i is not conversing with user 1, where r_i is the expected fraction of the conversation that user i is transmitting, and its value is given in Equation (4).

Let us now examine the algorithm's ability to discriminate between the actual conversation partner of user 1, and some other user. As we will see, this depends on the ratios r_1 and r_i, and in general it is optimal when $r_1 \approx 1/2$, that is, when the two parties talk in equal proportions. Conversely, in the extreme situation that there are four streams consisting of two conversations in which only one user speaks per pair (i.e., $r_1 = 0$ or 1, and similarly for the other two parties) it is impossible to correctly distinguish the two conversing pairs. Therefore, values of r_1 that approach $1/2$ allow for more pattern variability and thus intensify distinguishability.

Assume that user 1 is conversing with user 2, and consider some other user i. Let us define the quantity

$$\begin{aligned} \text{Diff}(i) = \max\{&\mathrm{E}_\pi[\text{Cim-asym}(1,2)] - \mathrm{E}_\pi[\text{Cim-asym}(1,i)], \\ &\mathrm{E}_\pi[\text{Cim-asym}(2,1)] - \mathrm{E}_\pi[\text{Cim-asym}(2,i)]\}. \end{aligned}$$

Note that we define this quantity only for the sake of analysis; the algorithm does not know it since it does not know that user 2 is the one who converses with user 1. In words, Diff(i) gives the difference between the cimilarities of the actual partner of user 1 (i.e., user 2) and user i (or the corresponding value for user 2 since the cimilarity function that we consider is asymmetric; whichever is higher). High values of Diff(i) indicate that either user 1 or user 2 is able to deduce that user i is not the actual partner. In Figure 18 we show how this difference behaves for various values of r_1 and r_i. What we can see, is that assuming that each user talks at least 10% of the time, there will be a difference of at least 0.15. If we assume that each user speaks at least 30% of the time then the difference is at least 0.29, and under the assumption that all users speak about the same time, the difference is at least 0.35. Hence, we can decide that user 1 converses with user 2.

Now we show that the empirical averages, which give the cimilarity measures that our algorithm computes, are close to the expected values. Since the summands are not independent random variables, we cannot employ a Chernoff bound. Nevertheless, it turns out that since the Markov chain converges fast we can obtain similar exponential bounds. This is formalized by some recent results that make use of the spectral properties of the Markov chain, or make use of information-theory tools (e.g., [21, 32]). The following theorem is found in [21].

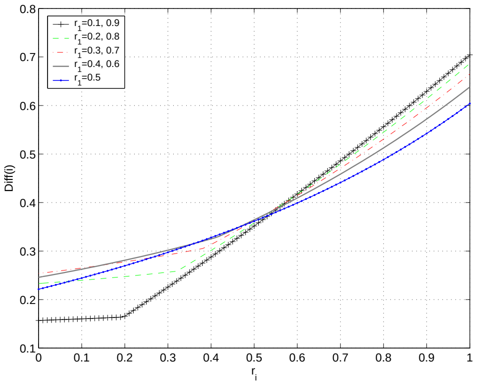

Figure 18: The difference of the gaps between the actual partner of user 1 (or user 2) and some other user i, for different values of the ratio r_1, for $\theta = 0.1$.

THEOREM 7.2. *Let $X_1, X_2, \ldots, X_n$ be a sequence generated by a time-reversible finite Markov chain with eigenvalue gap ϵ starting from an initial distribution q. Then*

$$\mathbf{Pr}\left(\left|\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}_S(X_t) - \pi(S)\right| > \beta\cdot\pi(S)\right) \le \left(2 + \frac{\beta\epsilon\cdot\pi(S)}{5}\right)\cdot N_q \cdot e^{-\frac{1}{20}\pi(S)^2\epsilon\beta^2 T},$$

where $N_q = \left\|\frac{q}{\sqrt{\pi}}\right\|_2$.

We can apply the theorem for our scenario, using $T = c\cdot\ln n$ for an appropriately chosen constant $c = c(p_1, p_i, q_1, q_i, \theta)$, as follows: The Markov chain is time-reversible, and the eigenvalue gap ϵ is a constant (which also depends on $p_1, p_i, q_1, q_i, \theta$). By application of Theorem 7.2, we get that for

$$T = c\cdot\ln n = \frac{60}{\pi(\Sigma_0)^2\epsilon\beta^2}\cdot\ln n,$$

where $\pi(\Sigma_0) = \min\{\pi(\Sigma_1), \pi(\Sigma_2)\}$, we have

$$\mathbf{Pr}\left(\left|\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}_{\Sigma_1}(X_t) - \pi(\Sigma_1)\right| > \frac{\beta}{3}\cdot\pi(\Sigma_1)\right) = o\left(\frac{1}{n^3}\right),$$

and that

$$\mathbf{Pr}\left(\left|\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}_{\Sigma_2}(X_t) - \pi(\Sigma_2)\right| > \frac{\beta}{3}\cdot\pi(\Sigma_2)\right) = o\left(\frac{1}{n^3}\right),$$

for sufficiently large n, if we assume that the values r_1, r_i, and θ are constant with respect to n. This implies that

$$\mathbf{Pr}\left(|R_i(T) - \mathrm{E}_\pi(R_i)| > \beta\cdot\mathrm{E}_\pi(R_i)\right) = o\left(\frac{1}{n^3}\right).$$

Essentially, we have shown that the computed cimilarity between users 1 and i is within β of its expected value, with high probability. By employing a union bound we finally deduce that after $T = c' \cdot \ln n$ time steps (with c' equal to the maximum value of c among all user pairs) the cimilarities for all the $\binom{n}{2}$ user pairs are close to their expected (under the stationary measure) values with probability $o(1/n)$. $\square$

Summarizing the findings of our analysis:

- Using the presented complementary similarity measures, we will eventually pair all streams with very low probability of error.
- The cimilarity measures converge exponentially fast to their expected values.
- As a consequence, the number of time steps required to match all the streams increases only logarithmically in the number of streams.

The third conclusion is particularly important, because it hints on the pairing performance for a large number of streams. We illustrate an example of the logarithmic increase in the convergence rate in Figure 19 using the previously used switchboard conversation data. Here, we plot the number of time steps required to achieve an accuracy of 90% as the number of users increases. We repeat the experiment 10 times with different sets of voice conversation for each stream cardinality and on the figure we report the average execution time over the 10 runs. One can easily distinguish a logarithmic trend.

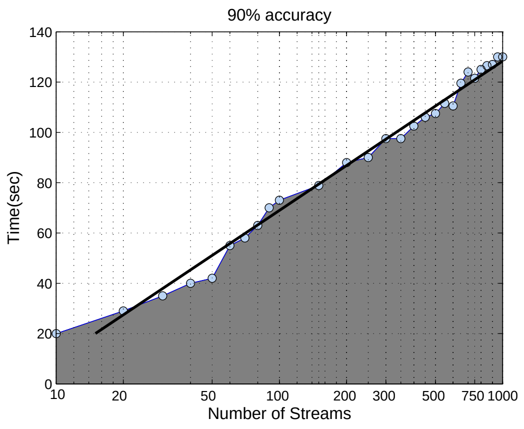

Figure 19: The number of time steps required in order to be able to achieve accuracy 90%, as a function of the number of users (linear interpolation of the points also shown). x-axis is on logarithmic scale.

Concluding comments on the analysis: Some of the constants that appear in the proof are a result of applying the general theory on Markov chains, and so they are an overestimate. In practice, one should still experience an exponential convergence rate albeit with lower constants than the ones provided by the analysis.

Note also that although our generative model for the conversations is fairly general and realistic, one can consider even more generic scenarios. For example, in [36], in addition to the two-state model presented here, the authors consider richer Markovian models; our results can be extended for those cases as well. More interestingly, one can consider more general stochastic processes, where, for example, the probabilities p_i or the noise θ might change over time. Although in this case we cannot model the system as a Markov chain, stochastic domination arguments can show that our results carry in those cases, as long as the streams of non-conversing users and the noise remain mutually independent.

8. EXPERIMENTS

As our experimental testbed we used real telephony conversations from switchboard data [23], which contained 500 pairs of conversations for a total of 1000 voice streams and consisted of multiple pairs of users conversing on diverse topics. Such datasets are typically used in many speech recognition contests for quantifying the quality of different speech-to-text processes. The specific dataset that we used, consisted actually of quite noisy conversational data and the length of each conversation is 300 sec. The original voice data have been converted to VoIP packets (using the protocol described in the VoIP section), then fed onto a local network using our custom made workload generator and recaptured by the planted data sniffers.

8.1 Comparison of Cimilarity Measures

In this initial experiment we compare the pairing accuracy of the three presented complementary similarity measures. We utilize the hard clustering approach which does not leave any unassigned pairs, therefore it introduces the largest amount of incorrectly classified pairs. However, since the hard clustering follows a more aggressive pairing strategy, this experiment essentially showcases the best possible convergence rate for the various complementary similarity measures.

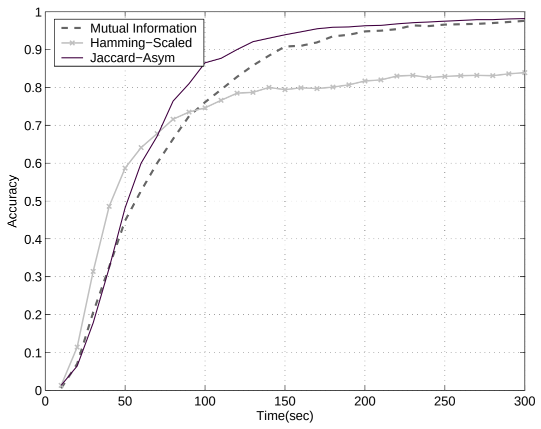

Figure 20: Pairing accuracy between 3 measures.

Figure 20 presents the pairing accuracy of the Mutual Information (MI), Jaccard Asymmetric and Scaled Hamming measures. Every 10 seconds we calculate the accuracy of the hard clustering algorithm by pairing each of the voice streams with the stream that depicts the maximum complementary similarity. Notice than in this way we do not necessarily impose a 1-to-1 mapping of the streams (hence, a stream may be paired with more than one streams). We report the results using the 1-to-n mapping, since we discov-

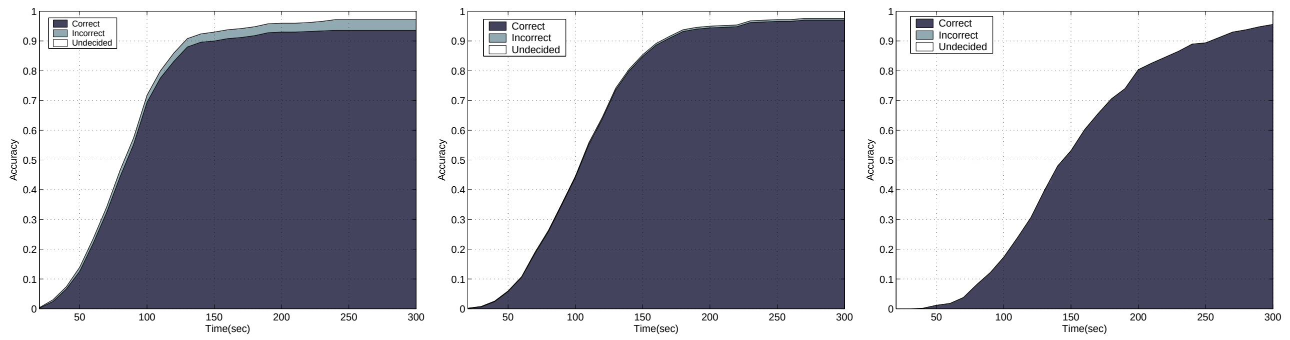

Figure 21: Progressive pairing for Asymmetric Jaccard. Left: $f = 1/2$, Middle: $f = 2/3$, Right: $f = 1$

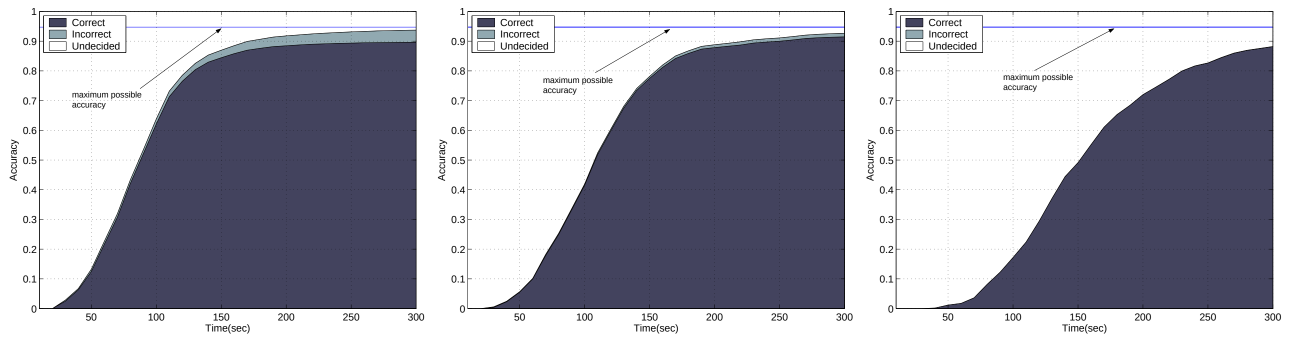

Figure 22: Progressive pairing for Asymmetric Jaccard using 50 singleton streams (their matches are not in the pool of monitored streams). Left: $f = 1/2$, Middle: $f = 2/3$, Right: $f = 1$

ered that it consistently achieves more accurate results than the 1-to-1 mapping.

On the figure we can observe that the Asymmetric-Jaccard measure is the best overall performer. It achieves faster convergence rate than the Mutual Information (90% accuracy after 120sec, instead of 150sec for the MI) and also a larger amount of correctly classified pairs at the end of the experiment. The Scaled-Hamming measure appears to be quite aggressive in its pairing decisions in the beginning, but flattens out fairly quickly, therefore it cannot compete in terms of accuracy with the other two measures. Since different measures appear to exhibit diverse convergence rates, as possible future work it would be interesting to explore the possibility of alternating use for the various measures at different stages of the execution, in order to achieve even faster pairing decisions.

In general, the results of this first experiment are very encouraging, since they indicate that the use of simple matching measures (like the Asymmetric Jaccard) can achieve comparable or better pairing accuracy than more complex measures (such as the Mutual Information). For the remainder of the experiments we will focus on the Asymmetric-Jaccard measure, and specifically on its performance using the progressive pairing algorithm.

8.2 Progressive Clustering Accuracy

The progressive algorithm presented in the paper has two distinct advantages over the hard clustering approach:

1. It avoids the continuous pairwise distance computation by leveraging the progressive removal of already paired streams.
2. It eliminates almost completely the incorrect stream pairings.

The second goal is achieved by reducing the aggressiveness of the pairing protocol, which in practice will have a small impact on the convergence rate (compared to the hard clustering approach). Recall that the progressive algorithm classifies the stream with the maximum cimilarity value (max_1) as a match, if

$$\max_1 - \max_2 > f \cdot (\max_2 - \text{cMass}).$$

The value f essentially tunes the algorithm's convergence rate. Smaller values of f mean that the algorithm is more elastic in its pairing decisions, hence achieving faster convergence, but possibly introducing a larger amount of incorrectly classified pairs. By imposing larger f values, we restrict the algorithm in taking more conservative decisions. This way fewer mistakes are made, at the expense of more prolonged convergence time.

Figure 21 presents the accuracy of the Asymmetric-Jaccard using values of $f = 1/2, 2/3, 1$. The darker part of the graph indicates the correctly classified pairs, the medium gray the incorrect pairings, and the white part are the remaining streams for which no decision has yet been made. From the graph, one can observe that for the examined dataset, $f = 2/3$ represents the best compromise between convergence rate and false pairing rate. The final pairing results after 300sec are: correctly paired $= 972$, incorrectly paired $= 6$, undecided $= 24$. Contrasting this with the hard clustering results at 300sec (correctly paired $= 982$, incorrectly paired $= 18$), we see that we can achieve fewer false assignments, while being quite competitive on the correct assignments and

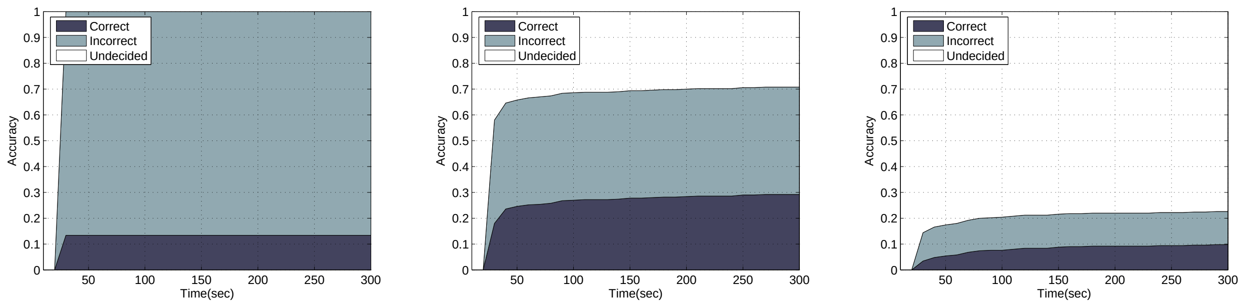

Figure 23: Progressive pairing for Asymmetric Jaccard using the criterion $max_1 > k \cdot cMass$**. Left:** $k = 1.5$**, Middle:** $k = 2$**, Right:** $k = 2.5$

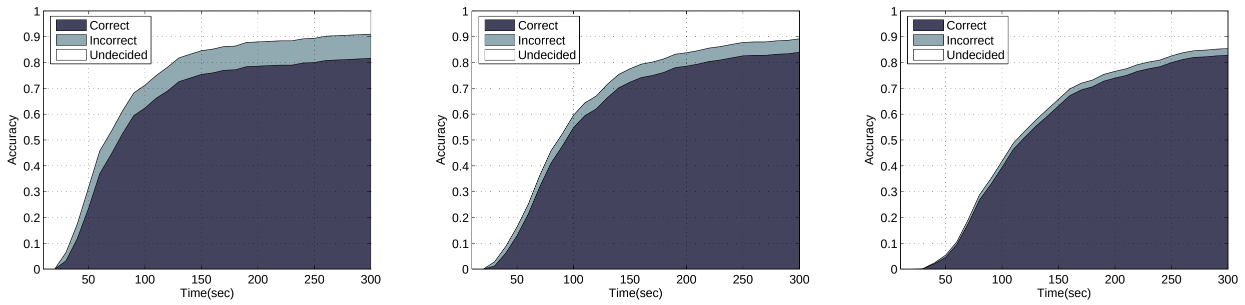

Figure 24: Progressive pairing for Asymmetric Jaccard using the criterion $max_1 > c \cdot \sigma + cMass$**. Left:** $c = 4$**, Middle:** $c = 4.5$**, Right:** $c = 5$

at the same time accomplishing a progressive clustering that is computationally less demanding.

Comparison with other clustering approaches: We also compare our previous stream pairing criterion scheme against two other approaches:

- The first criterion of comparison is the following:

$$\text{max}_1 > k \cdot \text{cMass}.$$

- The second one assumes a gaussian distribution of cimilarity values and attempts to find outliers in such a data distribution, therefore the matching criterion is the following:

$$\text{max}_1 > c \cdot \sigma + \text{cMass}.$$

  where σ is the standard deviation of the cimilarity values, after excluding the cimilarity value of the best max (i.e. the $max1$ value).

The clustering accuracy for these two pairing criterions are depicted on Figures 23 and 24 for different parameter values. We observe that the first criterion exhibits very low matching accuracy. The second criterion presents matching accuracy that reaches 85%, but it is still lower than our pairing criterion, which performs a better separation of the best candidate match against all the remaining streams.

8.3 Clustering Accuracy and Singleton Streams

In the previous experiment we had at our disposal both pairs for all conversations. Now we examine whether accuracy is impacted by the presence of singleton streams, that is, streams that have no homologue voice streams (possibly because it was routed through a different network router without sniffing capabilities). Now, we remove 50 random voice streams and we transmit only the remaining 950 ones. Each of these 50 streams correspond to one-way of a conversation. Hence, out of the 950 streams that are ultimately captured, 50 conversations cannot be correctly paired. The maximum possible streams that can be correctly identified is 900.

The results for these experiments for the different values of f are provided in Figure 22. Since now we cannot possibly achieve 100% accuracy (due to the singleton streams) we also indicate on the graph the maximum achievable accuracy (900/950). We repeat the experiment 10 times, removing each time a different set of 50 streams from the pool of 1000. The graphs indicate the average accuracy over the 10 runs of the algorithm. The results are almost identical as in the scenario without any singleton streams, indicating the resilience of the algorithm, which correctly did not pair the majority of the singleton streams with the remaining voice conversations.

8.4 Resilience to Latency

We conduct experiments which indicate that the matching quality is not compromised by potential end-to-end network delay. For simplicity of exposition we assume an end-to-end delay for each stream that remains constant as time passes (even though on a real network, delay will vary over time).

For this experiment we assume that each stream experiences a different global latency, drawn randomly from a uniform distribution within the range $[0, 2\delta]$, where δ is the observed one-way network latency. We conduct 4 sets of experiments with values $\delta = 40, 80, 160, 240$msec, therefore

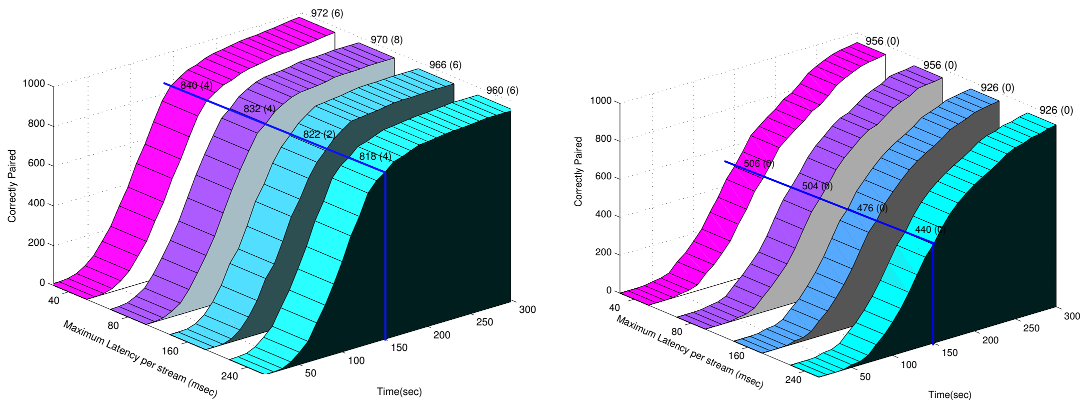

Figure 25: Accuracy of progressive pairing under conditions of end-to-end network delay. The graph depicts the correctly paired streams, while in the parenthesis we provide the number of incorrectly paired ones. Left: $f = 2/3$, Right: $f = 1$

the maximum possible synchronization gap between 2 pairing streams can be up to 2δ.

Figure 25 displays the pairing accuracy using the two clustering parameters that produce the least amount of misclassifications, $f = 2/3$ and $f = 1$. The 3D areas indicate the number of correctly paired streams, while on top of the surface we also indicate in parenthesis the number of incorrect pairings. We report the exact arithmetic values for the mid-point of the experiment (150sec) and at the end of the experiment (300sec). Generally, we observe that the clustering approach is robust even for large end-to-end latency. The accuracy of the pairing technique is not compromised, since the number of misclassified pairs does not increase. For latency of 40–80msec, the correctly classified pairs still remain approximately $970/1000$. This number drops slightly to $960/1000$ for 240msec of latency, but still the number of misclassified pairs does not change. Therefore, latency affects primarily the *convergence rate*, since ambiguity is increased, however accuracy is not compromised.

One can explain these results by noting that complementary similarity is most dominantly affected by the long speech and silence segments (and not by the very short ones). The long speech and non-speech patterns between conversing users are not radically misaligned by typical network end-to-end latencies, therefore the stream similarities in practice do not deviate significantly from their expected values.

Summarizing the experiments, we have shown that the progressive algorithm can achieve pairing accuracy that reaches 96–97%, while it can be tuned for faster convergence or minimization of false classifications. Singleton streams also do not penalize the algorithm accuracy. More significantly, we have demonstrated that the clustering performance is not affected by the network latency, since latency does not significantly affect the dominant temporal dynamics between conversational patterns.

9. SYSTEM DEPLOYMENT

We provide a brief description of a real system deployment of the conversation pairing algorithm, which is executed on top of the stream processing middleware (System S) [30] developed at IBM Research.

The stream processing core (SPC) middleware represents an API implementation for effective parallel execution of operators on streaming data, which is executed on top of a grid of parallel connected CPUs (Blade Servers). The basic notion of SPC is that of a processing element (PE). Each PE can be considered as an operator on a stream, which performs a specific task, accepting input from other PE's and sending output to some others. Therefore, the execution of a complex algorithm can be visualized as a processing graph, where PE's define their input and output.

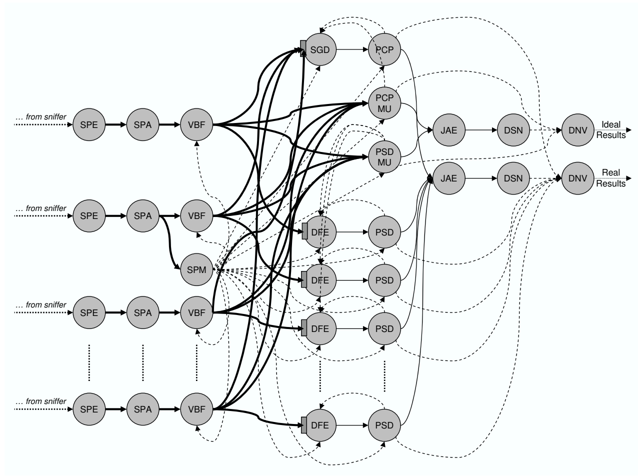

Figure 26: Stream Processing Graph instantiated in our stream processing unit: The nodes are the application operators and the edges represent the streams. The application operators include feature extraction, speaker detection, stream pairing, windowed join, etc. and are running on various, possibly remote, processing devices.

We implemented the progressive conversation pairing (PCP) algorithm as a processing element on top of the SPC middleware. The PCP is just a portion of a full scale application which performs real-time speaker identification and conversation pairing over VoIP conversations. The stream processing unit is fed by various remote sniffers, each carrying several compressed speech signals. These streams flow through a processing graph such as the one illustrated in Figure 26. The nodes correspond to PE's (stream operators) and the edges represent the streams. The processing elements include fast feature extraction from compressed speech signals (DFE), progressive speaker detection in quantized feature space [28] (PSD), stream binarization (SGD), progressive stream pairing of conversing parties (PCP). The results of the speaker detection and the conversation pairing are joined (JAE) in order to obtain both the conversing pairs and the identities of the speakers. Finally, the results are improved through techniques of probabilistic denoising (DSN).

The complete processing graph consists of 54 processing elements (PEs) running on 21 distinct computer nodes (Figure 26). The complete system can support real-time progressive detection of 300 conversations (600 streams), which effectively translates into processing of more than 10,000 speech frames per second. In order to support this high input load, the more complex operators (like the speaker detection [19]) needed to be replicated at multiple nodes. The conversation pairing element (PCP) given its progressive computation nature was particularly lightweight and therefore could handle the load on a single CPU.

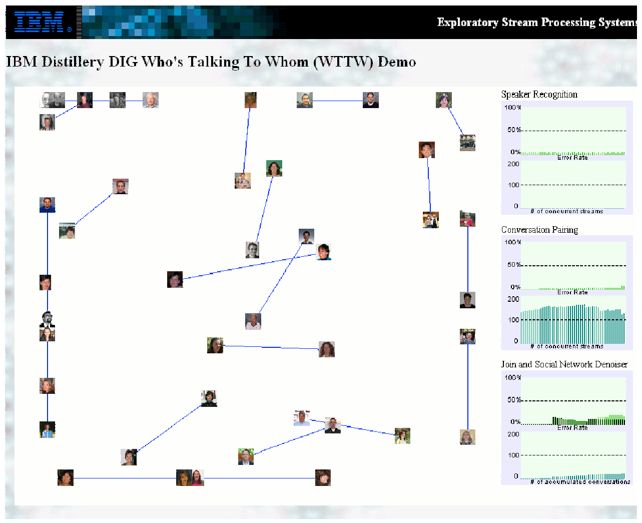
IBM
Exploratory Stream Processing System
IBM Distillery DIG Who's Talking To Whom (WTTW) Demo
Speaker Recognition
Error Rate
# of concurrent streams
Conversation Pairing
Error Rate
# of concurrent streams
Join and Social Network Denoiser
Error Rate
# of accumulated conversations

Figure 27: The visualization interface of the system deployment

In figure 27 we provide a snapshot of the graphical user interface that displays the system results. The main window provides a visualization of the speakers and conversations that are identified so far. Detected speakers are indicated by the existence of their photograph on the main window, and detected conversations are portrayed as lines. Pictures are connected with lines when both speakers and conversations are correctly discovered. The right part of the window captures the various system statistics, such as error rate and stream load for the speaker detection, the conversation pairing and the denoising elements. The interested readers are encouraged to watch a video demonstration of the system deployment and the presented GUI. [5]

10. DISCUSSION AND POTENTIAL SOLUTIONS

[5] `http://www.cs.ucr.edu/~mvlachos/voice/demo.wmv`

We have shown that the pairing of anonymous conversations is made possible because one can reconstruct the speech-silence signature of each voice stream through exploitation of the VAD in the VoIP protocol. Therefore, one could envision several ways of "tricking" the protocol into constantly sending all voice packets. We elaborate on various issues that may arise in those situations:

- First, one could turn off the VAD on the client side, hence forcing the client into sending all voice packets. This is an ability that may be provided by the telecom or VoIP provider company. While this seems as an obvious or direct solution, this is something that the providers might not be willing to do. We point out VAD is an inherent mechanism of many voice protocols because of the tremendous bandwidth savings that can lead to. Studies show that an average person speaks only 40–60% of the time during a telephone conversation [49, 6, 50]. These numbers are also verified in our dataset; the percentage of silence is very high reaching 59.2831%. Therefore, sending only the voice packets reduces the average bitrate and enhances overall coding quality of speech. In cellular radio systems (for instance, GSM and CDMA systems) based on Discontinuous Transmission (DTX) mode, this facility is essential for enhancing the system capacity by reducing co-channel interference and power consumption in portable digital devices [18, 20, 7]. In order to mask the complete absence of the "silent" packets, Comfort Noise Generation [54] techniques are typically deployed, which provide the impression of continuous conversation. However, Comfort Noise (CN) RTP packets can be easily detected and removed. Recall that the RTP Header of VoIP packets is not encrypted. CN RTP packets are indicated by a Payload type of 13, which means that an 8 KHz CN codec has been utilized (for more details see also IETF RFC 3389, at `http://www.rfc-editor.org/rfc/rfc3389.txt`).

  Because of the above, it is unlikely that the VoIP protocol will abandon the VAD feature. Comfort Noise packets can be easily filtered out and the binary speech-silence sequence can be thus still easily constructed. While certain carriers may provide the ability of disabling VAD (eg. Vonage), one can ask the question why would a VoIP provider let some people use much more bandwidth than others (yet pay the same bill)?

  For the above reasons this scenario is considered unlikely to happen.

- The second way for constantly sending packets, is to include a background sound source, that is above the VAD level (e.g. play music on the background while talking on the phone). This could trick the VAD into sending all the packets. It is indeed a plausible scenario and could be utilized as a defense mechanism. Implementation of this approach could potentially require a significant amount of reverse engineering and technical expertise, since the VAD mechanism is designed to distinguish the difference between 'speech' and 'non speech'. That is, between speech and {silence, noise, etc.}. The fact that VAD levels are adaptive is also expected to make the process even more challenging. This, however, raises also the question of whether sound quality has to be compromised or latency will be introduced in the system. For example, if someone's bitrate is too high, then the respective VoIP packets might be given a lower priority than someone else's with lower bitrate. In any case, a source sending a substantial amount of additional packets is expected to have a larger latency in the communication, which one might (or might not) be willing to accept.

11. ADDITIONAL APPLICATIONS

While in this work we have presented how complementary stream clustering could be utilized in pairing anonymous VoIP conversations, there are a number of other direct or indirect applications where this methodology could also be useful:

1) The coordination measures of a conversation can be utilized in assessing the quality of an online VoIP conversation [48, 25]. For example, if the complementary similarity in a two-way conversation is not above a properly defined threshold, then this might provide an indication of problematic communication between the speakers, therefore hinting on the presence of network latency, dropped packets, or other network problems. Therefore, the coordination measures can provide a level of assurance regarding the quality-of-service (QOS) for an online communication.

2) Consider a room with n people, where microphones are placed at different locations recording the on-going conversations. While the microphones can capture multiple conversations based on their placement, if the room contains at least n (properly situated) microphones, then Independent Component Analysis techniques (ICA) can be used to effectively unmix the original sources, and therefore reconstruct the original 1-way communications of each person [33, 27]. Afterwards, techniques like the one described in the paper, can determine the complementary 1-way communications, and therefore help assemble the original conversations by recombining the separated speech of each person.

3) The online complementary similarity measures that we put forward can also find great utility in generic load-balancing problems [56, 8, 26, 35]. Such problems arise very naturally in client-server architectures and the objective is to identify clients with complementary demand patterns. This is very useful because it allows the grouping of clients and their assignment on a single server, hence maximizing the utility time of the system, or conversely reducing its idle time. For example, in a computer grid where multiple processes are being executed, the objective is to discover processes/threads with complementary usage pattern (CPU/ hard-disk usage) that can be placed under the same computing space. This situation can be quite common for processes that wait the results of other processes, and therefore rarely exhibit overlapping executions. Other typical scenarios of load-balancing can be encountered in a variety of network applications, where the intention is to assign clients with complementary packet rates to the same server. Load balancing in a streaming environment has also been of great interest recently [55, 53, 45]. While in this paper we only consider pairs of complementary stream patterns because of the focus of this work, complementary pairing of multiple stream is also possible, which could effectively handle the aforementioned problems.

12. CONCLUSIONS

We have presented results indicating that intercepted VoIP data can potentially reveal private information such as pairs of conversing parties. Careful analysis of the voice packets coupled with an effective complementary pairing of voice activities can achieve high accuracy rates. We have also demonstrated that data encryption schemes cannot throttle the pairing of conversations. Many avenues are still open for investigation. Areas that that we are currently exploring are the provision for distributed execution [47] of our pairing algorithm, as well as the alternating usage of multiple distance measures at different execution stages of the algorithm. The ultimate objective of such efforts are to provide a pairing algorithm that exhibits fast convergence, in addition to being robust and accurate. We believe that our algorithms and pairing models could be of independent interest, for general pairing of binary streaming data. Closing, we would like to point out that the main objective of this paper was not to suggest ways of intercepting VoIP traffic for malicious reasons, but merely to raise the awareness that privacy on Internet telephony can be easily compromised.

13. REFERENCES

[1] C. Aggarwal, J. Han, J. Wang, and P. Yu. A Framework for Clustering Evolving Data Streams. In *Proc. of VLDB*, 2003.

[2] C. Aggarwal, J. Han, J. Wang, and P. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In *Proc. of VLDB*, 2004.

[3] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining Variance and k-Medians over Data Stream Windows. In *Proc of PODS*, 2003.

[4] S. Basu. *Conversational Scene Analysis*. PhD thesis, Massachusetts Institute of Technology, 2002.

[5] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The secure real-time transport protocol (srtp). In *IETF RFC 3711*, March 2004.

[6] A. Benyassine, E. Shlomot, H.-Y. Su, D. Massaloux, C. Lamblin, and J.-P. Petit. ITU-T Recommendation G.729 Annex B: a silence compression scheme for use with G.729 optimized for V.70 digital simultaneous voice and data applications. In *IEEE Communications Magazine, Vol 35(9): 64–73, 1997*.

[7] F. Beritelli, S. Casale, and G. Ruggeri. Performance Evaluation And Comparison Of Itu-T/Etsi Voice Activity Detectors. In *IEEE Signal Processing Letters, Vol. 9(3): 85-88*, 2002.

[8] A. Bestavros. Load Profiling: A Methodology for Scheduling Real-Time Tasks in a Distributed System. In *Proc. of ICDCS*, 1997.

[9] P. T. Brady. A Statistical Analysis of On-Off Patterns in 16 Conversations. In *Bell System Technical Journal, 47(1):73-91*, 1968.

[10] A. Z. Broder. On the resemblance and containment of documents. In *Proc. of the Compression and Complexity of Sequences*. IEEE Computer Society, 1997.

[11] A. Chakrabarti, K. D. Ba, and S. Muthukrishnan. Estimating Entropy and Entropy Norm on Data Streams. In *STACS*, pages 196–205, 2006.

[12] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better Streaming Algorithms for Clustering Problems. In *Proc. of ACM Symposium on Theory of Computing*, 2003.

[13] H. Clark and S. Brennan. Grounding in Communication. In *L. B. Resnick, J. Levine, & S. D. Teasley (Eds.), Perspectives on socially shared cognition*, pages 127–149, 1991.

[14] C. Clifton, M. Kantarcioglu, A. Doan, G. Schadow, J. Vaidya, A. K. Elmagarmid, and D. Suciu. Privacy-preserving data integration and sharing. In *DMKD*, pages 19–26, 2004.

[15] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan. Comparing Data Streams Using Hamming Norms (How to Zero In). In *IEEE Trans. Knowl. Data Eng. 15(3): 529-540*, 2003.

[16] G. Cormode and S. Muthukrishnan. Estimating Dominance Norms of Multiple Data Streams. In *Proc. of Annual European Symposium on Algorithms*, pages 148–160, 2003.

[17] P. Domingos and G. Hulten. A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering. In *Proc. of ICML*, 2001.

[18] R. Fulchiero and A. S. Spanias. Speech enhancement using the bispectrum. In *IEEE ICASSP, pages: 488–491*, 1993.

[19] G. N. Ramaswamy, J. Navratil, U. V. Chaudhari and R. D. Zilca. The IBM System for the NIST 2002 Cellular Speaker Verification Evaluation. In *IEEE ICASSP*, Hong-Kong, April 2003.

[20] G. Gabor and Z. Gyorfi. On the Higher Order Distributions of Speech Signals. In *IEEE Transactions. on Acoustics, Speech, and Signal Processing, Vol 36(4): 602–603*, 1998.

[21] D. Gillman. A Chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

[22] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. In *Communications of the ACM*, volume 42, February 1999.

[23] D. Graff, K. Walker, and A. Canavan. Switchboard-2 phase ii. LDC 99S79 – http://www.ldc.upenn.edu/Catalog/, 1999.

[24] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. In *IEEE TKDE, Vol. 15, No. 3: 515-528*, 2003.

[25] F. Hammer, P. Reichl, and A. Raake. The Well-Tempered Conversation: Interactivity, Delay and Perceptual VoIP Quality. In *IEEE International Conference on Communications (ICC)*, 2005.

[26] M. Harchol-Balter and A. B. Downey. Exploiting Process Lifetime Distributions for Dynamic Load Balancing. In *ACM Transactions on Computer systems 15:3*, 1997.

[27] A. Hyvrinen. Fast and Robust fixed point algorithm for independent component analysis. In *IEEE Trans on Neural Networks, 10(3): 626-634*, 1999.

[28] I. H. Tseng and O. Verscheure and D. S. Turaga and U. V. Chaudhari. Quantization for Adapted GMM-Based Speaker Verification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2006.

[29] J. Jaffe, L. Cassotta, and S. Feldstein. Markovian model of the time patterns of speech. In *Science, 144:884-886*, 1964.

[30] N. Jain, L. Amini, H. Andrade, R. King, Y. Park, P. Selo, and C. Venkatramani. Design, implementation, and evaluation of the linear road benchmark on the stream processing core. In *SIGMOD*, pages 431–442, 2006.

[31] M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *Proc. of SIGKDD*, pages 599–604, 2004.

[32] I. Kontoyiannis, L. A. Lastras-Montaño, and S. P. Meyn. Relative Entropy and Exponential Deviation Bounds for General Markov Chains. In *International Symposium on Information Theory*, 2005.

[33] T.-W. Lee, A. Bell, and R. Orglmeister. Blind source separation of real world signals. In *Proc. of IEEE International Conference Neural Networks*, pages 2129–2135, 1997.

[34] T. Li. A General Model for Clustering Binary Data. In *ACM SIGKDD*, 2005.

[35] D. S. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou. Process Migration. In *ACM Computing Surveys 32:3*, 2000.

[36] D. Minoli. Issues in packet voice communications. In *Proceedings of the Institution of Electrical Engineers, 126(8):729-740*, 1979.

[37] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming Data Algorithms for High Quality Clustering. In *Proc. of ICDE*, 2002.

[38] C. Ordonez. Clustering Binary Data Streams with K-means. In *8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 12 – 19, 2003.

[39] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming Pattern Discovery in Multiple Time-Series. In *Proc. of VLDB*, pages 697–708, 2005.

[40] T. V. Pham, M. Kepesi, G. Kubin, L. Weruaga, M. Sigmund, and T. Dostal. Time-frequency analysis for voice activity detection. In *Proc. of the 24th IASTED International Conference on Signal Processing, Pattern Recognition*, pages 244–249, 2006.

[41] R. V. Prasad, A. Sangwan, H. Jamadagni, C. M.C, R. Sah, and V. Gaurav. Comparison of voice activity detection algorithms for VoIP. In *Proc. of Interna-*

tional Symposium on Computers and Communications (ISCC), 2002.

[42] H. Sacks, E. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking in conversation. In *Language, 50*, pages 696–735, 1974.

[43] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. BRAID: Stream Mining through Group Lag Correlations. In *Proc. of SIGMOD*, pages 599–610, 2005.

[44] T. S. Saponas, J. Lester, C. Hartung, and S. Agarwal. Devices That Tell On You: Privacy Trends in Consumer Ubiquitous Computing. In *USENIX Security Symposium*, 2007.

[45] M. A. Shah, J. M. Hellerstein, S. Chandrasekaran, and M. J. Franklin. Flux: An Adaptive Partitioning Operator for Continuous Query Systems. In *Proc. of ICDE*, 2003.

[46] R. Sion, M. J. Atallah, and S. Prabhakar. Rights Protection for Discrete Numeric Streams. In *IEEE Trans. Knowl. Data Eng. 18(5)*, pages 699–714, 2006.

[47] J. Sun, S. Papadimitriou, and C. Faloutsos. Distributed Pattern Discovery in Multiple Streams. In *Proc. of PAKDD*, pages 713–718, 2006.

[48] A. Takahashi. Opinion model for estimating conversational quality of VoIP. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.

[49] H. Toral-Cruz and D. Torres-Roman. Traffic analysis for IP telephony. In *International Conference on Electrical and Electronics Engineering*, pages 136–139, 2005.

[50] C. Un and H. Lee. Voiced/Unvoiced/Silence discrimination of speech by delta modulation. In *IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 28(4): 398–407, 1980*.

[51] O. Verscheure, M. Vlachos, A. Anagnostopoulos, P. Frossard, E. Bouillet, and P. S. Yu. Finding ’Who is talking to whom’ in VoIP Networks. *Proc. of ICDM*, 2006.

[52] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the internet. In *ACM Conference on Computer and Communications Security (CCS)*, 2005.

[53] Y. Xing, J.-H. Hwang, U. Cetintemel, and S. Zdonik. Providing Resiliency to Load Variations in Distributed Stream Processing. In *Proc. of VLDB*, 2006.

[54] P. K. Yasheng Qian, Wei-Shou Hsu. Classified Comfort Noise Generation for Efficient Voice Transmission. In *Proc. of InterSpeech*, 2006.

[55] S. Zdonik, J.-H. Hwang, and Y. Xing. Dynamic Load Distribution in the Borealis Stream Processor. In *Proc. of ICDE*, 2005.

[56] S. Zhou. Performance Studies of Dynamic Load Balancing in Distributed Systems. In *PhD Thesis, UC Berkeley*, 1987.

[57] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *Proc. of VLDB*, 2002.

[58] P. Zimmermann. Zfone: `http://zfoneproject.com/`.

[59] Findnot – `http://www.findnot.com`.