2010 16th Asia-Pacific Conference on Communications (APCC)

Smarter Searching for a Network Packet Database

William D. Kenworthy
School of Information Technology
Murdoch University
Perth, Western Australia
Email: W.Kenworthy@murdoch.edu.au

***Abstract*—Bioinformatics has developed some very sophisticated algorithms used for searching within the large amounts of data produced by sequencing projects. Introduced here is a method that leverages these algorithms for processing computer network data traffic to find and classify information in a way that avoids many of the operations that are required of more traditional methods of processing network data packets. It can be used to avoid processing the data through a protocol stack in order to extract the content to make it ready for searching as well as being able to a access the advanced search capabilities developed for biological data. Raw data (for example whole Ethernet packets or strings/ motifs) can be directly searched for within a database of structure signatures to indicate packets of interest via structures within the query packet. The structures referred to here are the "patterns of patterns" within the data. This paper describes the results obtained by creating a searchable database based on bioinformatics techniques and searching it using sample data.**

I. Introduction

Searching for one or more items in a collection of items is a very common task in computing. For data traveling over a computer network, this can be particularly difficult. The method of searching using the approach pioneered for bioinformatics to process biological data is applicable to other forms of data having similar structure - in particular applying it to data traveling over a computer network opens up a number of new ways to find information hidden in large data flows. This paper documents the initial idea and feasability of using algorithms developed to target biological information on computer network data flows.

Searching the content of packetized network data (such as that carried over TCP/IP [1] networks) for keywords and terms is usually undertaken via extracting the content using the appropriate protocols for the data stream involved, the content of which is then searched for items of interest. Meta information about the protocols present is either available from the protocol stack processing the data stream, or is added via a database embedded in the application being used (for example, the WireShark [2] network protocol dissector/analyzer software). Most common search techniques use simple text matching or regular expression based search mechanisms, but these are less effective at the lower levels of the protocol stack due to fragmentation and interleaving of other data between parts of the message to be scrutinized, often in the middle of a word or phrase. It is the task of a protocol stack like that used for much of the worlds data communications (TCP/IP [1]) to reassemble the original data into the same format/package that was originally sent so search operations can be applied to it.

What we are proposing here is to avoid the computational load and resource usage of search operations at higher levels of the protocol stack by storing signatures or fingerprints representing known data and interesting terms that may be carried as part of a packet payload in a separate database. Any new packet of data from a networks can be searched within the database and the resulting areas of congruence (or motifs in common) ranked in order of significance giving generating meta data about the packet independent of the protocol stack. In this way particular terms and phrases can be highlighted without having to search for each one as a separate process. An added benefit of the algorithms being proposed is that they have some ability to deal with data that is interleaved with unrelated data or split across boundaries.

This differs from the usual process of "deep packet inspection" [3] which usually entails processing the packet data through a large part of the protocol stack to retrieve the (sub) data stream to target. An example of deep packet inspection is to search incoming web (using the HTTP protocol) query results for banned terms or information leakage. The program tasked with analyzing the content usually has to extract and format the HTTP sessions in order to scan the text.

For fast efficient search operations, some very sophisticated algorithms and programming techniques have been developed for bioinformatics tasks. Particularly important for the method being described here are both the very fast database lookups possible, and the statistics made available which enable sense to be made of the very complex relationships involved. There are a number of differences in the strategy being proposed here to what normally occurs in deep packet inspection. The basic process is:

1) For each whole packet of data from the network
2) Search the signature database using the whole packet
3) Process the possibly numerous motifs-in-common between the search packet and database entries using valid statistical methods
4) Select the appropriate "hit" on the database
5) Take the appropriate action

Many different types of database systems are available which are suitable for solving all sorts of problems. These range from simple key/value pair constructs, through to RDMS (Relational Database Management Systems) and OO databases (Object Oriented databases). The designers of the

978-1-4244-8129-3/10/$26.00 ©2010 IEEE

large databases that support bioinformatics use a number of different database types (National Center for Biotechnology Information - NCBI [4]) depending on need. The database format that is of interest here is used to both supply datasets to external researchers and as a means to efficiently, and speedily search the large datasets used in bioinformatics for statistically validated matches using poorly defined search terms. "Poorly defined" in this context has a meaning of data that is "nearby" as defined by an appropriate metric or distance figure. Databases and related indexes for DNA data are created using a software program known as "formatdb" [5] from the NCBI [4]. The databases created are in the form of indexed signatures that can be searched using algorithms such as BLAST [6] and Smith and Waterman [7] giving results ranked on the statistical importance of the matches. The important point to note is not the use of an indexed flat file database that is used for storage, but the search mechanism and statistical support for the results. BLAST [6] and the competing Smith and Waterman [7] algorithm (used by the fasta [8] series of programs) algorithms commonly used to search genomic data use methods that are effectively searching for structures contained within the data query against structures contained within the data signatures stored in the database. The method used by the BLAST [6] algorithm (via the software utility of the same name) uses a sophisticated word based comparison method to identify areas of congruence between the input vector and each record in the database. It builds up a set of motif's in the query vector that correspond to motifs within vectors in the database. It then calculates the quality score for each motif returning those above a threshold in the form of an report ordered by significance (based on each score).

It is possible to pre-format network data packets (such as TCP/IP data packets [1]) for use with formatdb [5] to produce a very fast, high performance database of packet types that allows fast lookup and classification of data packets with relevant meta-data concerning statistics relating to false alarm rate and identity. This allows not only a lookup of "what is the closest match", but "what else is a near match" along with statistically based rankings. While protocol information is important, it is the payload (packet content) that is of most interest in the use of this type of search. While it is quite practical (and IBM has succeeded in doing this for email [9]) to use similar algorithms being discussed here to search content recovered from a stream of network traffic, there is more information available before the low level information is stripped via the normal operation of network protocol stacks. Bioinformatics algorithms have been applied to parts of the computer security equation as demonstrated by Coull et. al. in 2003 [10] but not in the way being explored here at this level of the protocol stack.

II. Statistics

To find both exact and near matches for a query in the database, a way is needed to accurately rank the results taking into account the length of the query and the length of the matching parts of the query to one or more database entries. A method of statistically detecting false hits is also desired. A false hit is one that is returned by a query and ranked as significant when it is not. As databases get larger it becomes more likely that they will contain random matches to a motif in a query. The chance of this occurring can be calculated statistically and the resulting value is known as a "False Alarm Rate" where it can be defined for the use to which it is put here as " The chance of retrieving a false value from the database for a motif of this length in a database of this size." Unfortunately, a false alarm rate is an analog value, not yes/no categorization so it is up to the user to set a threshold value - the closer to zero the better. The NCBI Handbook [4] suggests an upper limit of the false alarm rate for biological comparisons to be considered significant is approximately - 0.01.

In bioinformatics, two types of statistics are used to calculate the rankings - a "score" and an "E-value". The score is used to rank the hits in the database according to their size and quality of the match while E-values are used to calculate the possibility that a result is false - the calculated False Alarm Rate based on the specific parameters for this query. A significant match in the database is one that is characterized as having a high score, and a low E-value (a good match with an expectation that it is due to a real data similarity rather than just coincidence.) Note that these values are analog in nature and do not represent yes/no decisions - they are usually managed using thresholds to remove obviously poor/wrong results (those with a high false alarm rate or very low score.) with the remainder being presented for human examination.

A. Score

The score is a value that is used to compare and rank the comparisons made by algorithms such as BLAST and FASTA. It is a composite metric that tries to provide a single figure that takes into account the similarity and size of individual matches. For BLAST, NCBI [4] gives the description: "The value S' is derived from the raw alignment score S in which the statistical properties of the scoring system used have been taken into account. By normalizing a raw score using the formula:

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

a bit score *S'* is attained, which has a standard set of units, and where *K* and *lambda* are the statistical parameters of the scoring system. Because bit scores have been normalized with respect to the scoring system, they can be used to compare alignment scores from different searches." [4].

The larger the score, the better the match in the quality of the *per character* match over the length of the match. The score is calculated after alignment has occurred - alignment being the process of *aligning* like motifs within both the query sequence and each sequence stored within the database. This may require gaps to be inserted in one or other of the sequences in order to move the motif's into a proper relationship - areas of high similarity being directly compared with other highly similar areas. The score is then calculated

as a distance metric using the above formula which calculates a higher value where there are areas with a high similarity between the query and database entry being compared. The formula takes into account both the length of the matching areas and the quality of the match enabling retrieval of the best and most relevant matches in the database along with a false alarm rate statistic. This allows likely bad results to be discarded and the remainder ranked from closest to most distant by similarity. The size of the query vector and the database entries does not matter - its the size of the motifs shared between the query and the relevant database entry that is important.

B. E-Value

E-Values (or Expectation-Values) are a useful method for calculating the significance of results returned from database searches. The E-value as used by BLAST [6] and other bioinformatics programs is usually quoted as "a parameter that describes the number of hits one can expect to see by chance when searching a database of a particular size." [4] BLAST uses the following formula to calculate the E-Value for two sequences of length m and n with a normalised score of S':

$$E = mn2^{-S'}$$

A value of zero, or very close to zero is desired which means that there is only an extremely small chance the result is due to the coincidence that a matching piece of data is present in the database. E-Values decrease exponentially with the score that is assigned to a match between two sequences. As the E-Values increase, the "false alarm rate" increases until results can be considered worthless. The actual value that is used to determine significance vs insignificance as a cut-off varies depending on the needs of researchers - in bioinformatics, using too stringent value can eliminate possibly relevant results (those that are more distantly related), while a too loose value will cause a researcher to waste time analyzing results that have no relevance.

III. METHOD

The results being discussed here were obtained using a manually constructed database consisting of anonymously generated network data. The source data was assembled by using an isolated island data network set up to specifically collect network data using tools such as WireShark [2] with volunteer users sending email, VoIP calls and various other network actions. Users are anonymous in that they are specifically created accounts on generically set up desktop computers and servers to send innocuous messages knowing that the contents must contain no personal or identifiable information about real people and therefore ensuring that no personnel/private data is collected. For example, the testers use random user names and content for emails sent over the network where the specification requires (for example) "10 emails to each of 4 recipients from an administration account on host 2". The collected data was stored in the pcap [11] format for post processing.

The data was then translated into a form suitable for processing by bioinformatics programs. In a project such as this which is using highly complex algorithms implemented in complex software by large institutions/teams of programmers, it is not practical for individual researchers to rewrite such software for specialized tasks. In this case the solution was to process the pcap [11] formatted data packets from the Wireshark [2] software that was used to collect and store the data for analysis into a form suitable for input to standard, unmodified bioinformatics programs such as BLAST [6] and clustalw [12]. Processing involved mapping the binary representation of the network data onto the DNA symbol alphabet used by bioinformatics and storing the result in FASTA [13] formatted files.

These files were then processed into a fast, high performance database by the NCBI formatdb [5] program. This is not a database of the packet data in the usual terms, but contains motif's and signatures used by the NCBI BLAST [6] (Basic Local Area Search Tool) suite of programs for high speed searching by advanced heuristic algorithms, the results of which are accompanied by statistics for validation. The BLAST [6] programs uses a sophisticated word based search algorithm to produce signatures from a search query that can be applied against a preprepared database of signatures such as that produced by formatdb [5]. The results returned by the search are a list of entries in the database ranked in order of score and qualified by E-value.

The default substitution matrix for nucleotide searches has been used in the tests being discussed here in order to avoid biasing the results. The default matrix for the software used is an "identity matrix" which gives a bias in favor of one to one matches between symbols. This one area where custom software would be needed to implement the feature. At this point in time no effort has been put into implementing or testing the use of substitution, but it is envisaged to be a useful extension for the future.

In bioinformatics, a qualified user (usually a domain specialist such as a biologist) will examine the data to make a decision on the data quality and usability - this can be automated here because:

1) network packet data is well characterized and not as mutable as biological data
2) We are only interested in identification (at this stage)

This allows a simple software routine to extract the top few significant matches and select the one with the highest likelihood of being correct. This will usually, but not always be the top of the list. For a biological search, a biologist would examine the list and select the most likely candidate based on domain knowledge. For the proof of concept testing undertaken for this project, the top of the search results was always the closest match to the query packet. This can be expected to change when motif or partial packet searches are tested. Further to this, it was identified that the simple databases created for testing this method included information such as IP numbers which may be expected to influence full packet queries but not motif based searches.

This method has produced a set of searchable data in the form of a database formatted for searching by the BLAST [6] utility and tools that can take an unknown packet of data and search it against the database. The tools take the form of simple Perl scripts used to extract and format the data for a single packet from a pcap capture file - this is then used to search the database using the BLAST [6] program.

The database is created from a FASTA formatted [13] source file containing each of the samples desired in the database. The "formatdb" [5] command is used as in the example below to create the component files for the database - in this case "MyDatabase.nhr, MyDatabase.nin, MyDatabase.nsq". To create these files use the following invocation of:

$$formatdb\ \textit{-p F -i}\ MyDatabase.fa \tag{1}$$

The next command is used to search the database using the prepared packet data in the file packet.fa, returning the top ten matches.

$$blastall\ \textit{-v 10 -p blastn -d MyDatabase -i}\ packet.fa \tag{2}$$

The result of this process is a file containing the hits in the database along with supporting statistics.

IV. Results

Using the above commands, the contents of the file packet.fa are used to search the prepared database. The search query used for this demonstration is a typical lookup sent to a DNS server. Only the first (top) ten results (specified as a command line argument) are used here as they will represent the top matches between the search query and the database. The results are returned as a plain text file with multiple parts.

The BLAST programs [6] results file is complex being designed for the requirements of bioinformaticians, but for the purposes of searching for wanted motif's (terms of interest) carried by network data packets it is ideal for testing as it clearly shows all the components used in calculating the various statistics. In particular the section "Sequences producing significant alignments" lists the top ten (as asked for in this search) results along with statistical validation. The larger the score and the smaller the E-value the better the match and match quality. A typical "Top Ten" list with related statistics for each item is shown in Figure 1 and following this are sections displaying the actual local matches for each of the top ten results with supporting statistics, an example of which is shown in Figure 2.

The local match sections show in detail the individual motif's in common between the query and each database entry. The top line is the "FASTA comment line" which in this case contains metadata from the Wireshark [2] capture program - this would not normally be available in a production system but is useful for testing. Next comes the length of the query vector in characters. Next is the calculated Score and Expect Values. Note that this is for the motif only, not the whole vector. In the example shown here its calculated to be 165 bits for a motif of 83 characters long as a 100% match with an E-value of $3e^{-42}$ which means quite a low chance of the result being a false. The following entries in Figure 1 have a lower score and not as close a match to the query. Any characters which do not match are shown in context in the next section of Figure 2 though in the example given here the match is 100%. The above information is repeated for every motif in the query identified in the database.

```
                                                                    Score     E
Sequences producing significant alignments:                         (bits)  Value

6747 603.350485 192.168.1.104  192.5.5.241  DNS Standard query A...   165   3e-42
3538 321.187936 192.168.1.21  192.168.1.104 DNS Standard query A...   157   7e-40
3317 299.994486 192.168.1.21  192.168.1.104 DNS Standard query A...   157   7e-40
6485 581.725795 192.168.1.104  192.36.148.17 DNS Standard query ...   157   7e-40
3392 308.618746 192.168.1.104  193.0.14.129 DNS Standard query A...   157   7e-40
3557 322.181778 192.168.1.21  192.168.1.104 DNS Standard query A...   157   7e-40
6525 586.538079 192.168.1.104  193.0.14.129 DNS Standard query A...   157   7e-40
3622 328.988543 192.168.1.104  192.168.1.21 DNS Standard query r...   157   7e-40
3537 320.618817 192.168.1.104  199.7.83.42  DNS Standard query A...   157   7e-40
3479 315.818991 192.168.1.104  192.203.230.10 DNS Standard query...   157   7e-40
```

Fig. 1. Top ten packets matching query with supporting statistics.

```
>6747 603.350485 192.168.1.104  192.5.5.241  DNS Standard query A wwwcache...
          Length = 332

 Score =  165 bits (83), Expect = 3e-42
 Identities = 83/83 (100%)
 Strand = Plus / Plus


Query: 94  cacgtagggactaaagggaaaaccggataaaaaccaaccttacgatggcccaaaaatcca 153
           ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 94  cacgtagggactaaagggaaaaccggataaaaaccaaccttacgatggcccaaaaatcca 153

Query: 154 aaaatacgagatctggttgtttg 176
           |||||||||||||||||||||||
Sbjct: 154 aaaatacgagatctggttgtttg 176
```

Fig. 2. Motif from query and an equivalent database match.

V. Discussion

The results obtained show that this method is a viable way to search network data streams for strings and motifs. Having a separate database of search terms and data types available for comparison allows the search process to be decoupled from the protocol stack.

However during the design and testing of this new method, a few issues became apparent. Network data traffic has some characteristics that are not too dissimilar to biological data in the way they cause problems for the algorithms and can be handled in a similar fashion.

A. Duplicate Data

Typical network data traffic has a large and constant supervisory component such as arp requests (Address Resolution Protocol [1]), heartbeat and keep-alive packets. These are usually simple, identical or nearly packets repeated at regular intervals which over time amount to a large number. If all packets collected including such duplicates are stored in a database, they create the same effect on the search process as low complexity regions and over represented features such as repetitive elements do on biological database searches. As they are over-represented, any statistical ranking of results containing such packets create distorted and biased statistics. Any search which included one of these common, over-represented packets or major parts thereof would return many (thousands) of results before any nearby matches would be

listed, undoing one of the large advantages of such a search strategy. In biology, the standard way of handing such features is to filter them out of the search query as in many cases biologically relevant data may still be obtained from duplicate data so it is usual to store every instance in the database and only search them if required. A more practical method for networking data is to allow only one sample of otherwise identical packets into the database.

Another, minor problem is that certain parts of the packet headers (such as source and destination IP numbers) are also present in many or most entries in the database giving rise to similar problems though with effects not as critical on the results as over-represented full size packets. A more serious issue is that any search will include many hundreds or more of small matches in the results file. These small matches can produce the equivalent of 200 pages or more of irrelevant data using significant resources in the process - this is a side effect of using software not designed specifically for this purpose and would be eliminated by using proper thresholds not available in this software.

B. Substitution

Substitution in biological terms is the replacement of certain amino acids with other amino acids of a similar chemical function. The replacement occurs semi-randomly and for biological applications, it can be empirically profiled to derive "substitution tables" which are a likelihood that a particular amino acid will be replaced by another particular amino acid. For the use of the algorithms being discussed here, it makes sense for certain operations to use substitution. A simple example is the substitution of a single character to indicate "$", "S" or "s" in the word "sex" in email Spam to improve detection. A substitution table mapping all variations to a single letter will enhance the Spam detection process. While in bioinformatics substitution is usually only required for single molecules (thus a single symbol), for the uses being discussed here symbol and word substitutions would be required. Rigoutsos and Huynh investigated this in 2004 with their "Chung-Kwei" Spam detection system. [9].

C. Alternative Distance Metrics

The algorithms used here are not the only ones that may be used to calculate a distance between two or more vectors of symbols. "Hamming distance" [14] calculations have been used for many years in communications and can also be used derive a distance metric. More work needs to be done to test and tune the various algorithms involved as the test implementation uses the standard bioinformatics algorithms for convenience as their characteristics appear to provide a good match to the typical network data profile.

VI. Conclusions

The process of creating and searching a database of network packet information using bioinformatics principles works well in principle. It is feasible to create a sample database containing representatives of many different types of common and uncommon packet types and content for identification purposes. Such a database can include examples of both good (or acceptable) data and bad data that is used for nefarious purposes. Detecting such bad data on a network could be used to flag their presence to an administrator or response program for action. There is also the possibility of searching for defined phrases and terms in the data stream without needing to process the packet through a protocol stack to get at the data at the application layer, providing flexibility and a possible saving in resource usage. The process used here to test and implement the described methodology is very inefficient if used for a production system but it has successfully proven the concept is workable. This and previous work has proven the viability of using the same type of algorithms designed to process biological data to process network data in a packet stream. Future work using a system based on the equivalent of the bioinformatics alignment process used in bioinformatics but implemented to specifically target network packet data shows great potential.

References

[1] DARPA, *RFC793: TRANSMISSION CONTROL PROTOCOL: PROTOCOL SPECIFICATION*, september, 1981 ed., Defense Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209, September 1981.

[2] G. Combes, "Wireshark," 2006. [Online]. Available: http://www.wireshark.org/

[3] M. Becchi and P. Crowley, "Efficient regular expression evaluation: theory to practice." San Jose, California: In Proceedings of the 4th ACM/IEEE Symposium on Architectures For Networking and Communications Systems, 2008, pp. 50–59. [Online]. Available: http://doi.acm.org/10.1145/1477942.1477950

[4] NCBI, "NCBI handbook," 2003. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=hand

[5] T. Tao, "Program parameters for formatdb and fastacmd- two blast database related tools," Apr 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/formatdb_fastacmd.html

[6] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool." *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, October 1990.

[7] T. Smith and M. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–7, 1981, msw-042.pdf.

[8] W. R. Pearson, "Rapid and sensitive sequence comparison with FASTP and FASTA." *Methods in enzymology*, vol. 183, pp. 63–98, 1990.

[9] I. Rigoutsos and T. Huynh, "Chung-kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (spam)." Proceedings of the First Conference on E-mail and Anti-Spam, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6804&rep=rep1&type=pdf

[10] S. Coull, J. Branch, B. Szymanski, and E. Breimer, "Intrusion detection: A bioinformatics approach." Proceedings of the 19th Annual Computer Security Applications Conference, 2003.

[11] Degioanni, L., Risso, F., and G. Varenni, "Pcap next generation dump file format," Mar 2004. [Online]. Available: http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html

[12] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. Gibson, D. Higgins, and J. Thompson, "Multiple sequence alignment with the clustal series of programs." *Nucleic acids research*, vol. 31, no. 13, pp. 3497–3500, July 2003.

[13] NCBI, "Description of the fasta format," Feb 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/blast/fasta.shtml

[14] D. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge, UK: Cambridge University Press, October 2003. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/itila/book.html