Three statistical approaches to sessionizing network flow data

Patrick Rubin-Delanchy†, Daniel John Lawson
Heilbronn Institute
School of Mathematics
University of Bristol
Bristol, UK
†Email: patrick.rubin-delanchy@bristol.ac.uk

Melissa J. Turcotte
Advanced Computing Solutions
Los Alamos National Laboratory
Los Alamos, US

Nicholas Heard and Niall Adams
Department of Mathematics
Imperial College London
London, UK
and
Heilbronn Institute, University of Bristol

***Abstract*—The network traffic generated by a computer, or a pair of computers, is often well-modelled as a series of *sessions*. These are, roughly speaking, intervals of time during which a computer is engaging in the same, continued, activity. This article explores a variety of statistical approaches to re-discovering sessions from network flow data using timing alone. Solutions to this problem are essential for network monitoring and cyber-security. For example overlapping sessions on a computer network can be evidence of an intruder 'tunnelling'.**

I. Introduction

Network flow data, including Cisco Systems' Netflow, are data collected by routers recording activity on a computer network. The data typically comprise a sequence of records, each called a *flow*, representing a communication (or some part of) between two computers. A flow contains a time-stamp denoting the start of the communication, a source internet protocol (IP), a destination IP, and further information about the communication (e.g. the ports used). Importantly, a flow does not retain any of the transferred data and so the routers' collection generates data of many orders of magnitude smaller than the amount of traffic on the network. Still, the widespread and indiscriminate use of the internet and internal network services in modern corporations and institutions creates huge network flow data stores. For example Imperial College London, a leading university in the UK with around 10000 students and 3000 academic and research staff, generates about 14 Terabytes of flow data in a month. For Los Alamos National Laboratory, a major US research institution specialising in science and technology for nuclear deterrence, the figure is roughly 30 Gigabytes a day.

For businesses and institutions the loss of confidential personnel records, intellectual property or information on business negotiations and strategy can be very damaging. Therefore protecting these data from malevolent outside actors is an important aspect of these organisations' cyber-security. A specific risk and recurring intrusion pattern is an attacker (say X) somehow compromising a computer on the network (say Y) and then proceeding from there to find valuable data on other machines — thereby bypassing firewall defences. The point is that Y is rarely the intended target but rather an opportunistically infected machine, for example by its user accidentally installing malware attached to an email. This pattern of attack is described in detail in [1]. In this strategy to perform any operation on a target computer (say Z) the attacker must typically create a remote session from X to Y and then another from Y to Z. This behaviour may be detectable in network data from the fact that the session from X to Y envelopes the session from Y to Z in the temporal sense, that is, the first session's start (resp. end) is earlier (resp. later) than the second's start (resp. end) [2]. Of course such an analysis presupposes that sessions can be identified in network flow data. This is the focus of the present article.

The Open Systems Interconnection model [3] provides a conceptual model for how computer networks and the internet operate, in seven layers of abstraction. To some extent network flow data can be seen as data collected at the fourth layer of abstraction, called the Transport Layer, whereas the concept of a session appears in the fifth layer of abstraction, the (aptly-named) Session Layer. However our definition of a session is somewhat more general. In this article we call a session any period of time where two computers engage in the same, continued activity.

This article explores a number of statistical techniques to perform sessionization, that is, the operation of extracting sessions from network traffic. To set out the parameters of the problem precisely, consider a sequence of flows generated by communications between a client computer X and a server Y. Over a period of observation $[0, T]$ we will assume that the start times of the flows form an (ordered) sequence of distinct *events*, $0 \leq t_1 < \ldots < t_n \leq T$. (These are treated as continuous random variables, even though in practice times can only be measured and stored at a finite resolution.) Our analysis focusses on sessionization *using only* those events. Typically there would be other clues in the sequence of flows, for example, in the TCP flags, the ports used, or the flow end times. However using this additional information also makes the task more complex because communication protocols and router collection policies differ. Hence as a preliminary investigation it is useful to understand how much can be achieved using only timing information.

The remainder of this article is structured into three main parts, corresponding to three statistical approaches to the problem. In Section II we investigate a simple time expiry policy, where sessions are started on events and closed if the time since the last event exceeds a certain threshold. In Section III we view the problem from a clustering perspective, and show that various ideas from the literature on spatial point patterns can be made relevant. Finally Section IV proposes a regime-switching process to model the data, and discusses some computational issues related to that approach.

II. A TIME EXPIRY POLICY

In this section we analyse the following procedure. Processing the events sequentially in time, a session is closed whenever the time since the most recent event exceeds a user-supplied threshold B, and a new session is opened on the next event. The pattern of open and close times provides a sessionization of the data.

One advantage of this procedure is that it is very well suited to real-time monitoring. Even when there is no requirement to process the data in real time, the fact that this can sessionize the data in one pass (assuming it is already time ordered) can be crucial when processing large data sets.

The procedure has one tunable parameter, the threshold B. The choice of how it should be set can be related to some concepts in statistical hypothesis testing. There, two forms of error are identified [4]. A type I error is committed if the null hypothesis holds, but is rejected. A type II error is committed when the alternative hypothesis is true, but we fail to reject the null hypothesis. If a session is closed prematurely then we have wrongly detected a change in activity, and so in a sense a type I error has been committed. Similarly failing to close a session when it has ended can be seen as a type II error.

Suppose that while a session is active, the inter-event times $\Delta_1 = t_2 - t_1, \Delta_2 = t_3 - t_2, \ldots$ (remember the events are flow start times) are independent replicates of a random variable Δ with known distribution. The temptation might be to choose B as a quantile of Δ. For example, following the hypothesis testing tradition we might choose B so that $\text{Prob}[\Delta \geq B] = 5\%$, thereby closing a session whenever the time since the most recent event is *significant* at the 5% level. A problem with this approach is that the rate of false closure is a function of the rate of events, not time itself, so that more active processes are opened and closed more often.

To address this issue it is useful to view the time since the most recent event as a stochastic process evolving in time, denoted $Z(t)$. An example of Z is shown in Figure 1b). The time-expiry policy closes a current session if Z crosses the boundary B and then opens a new session at the next event. Denote by s_i the current session start time as computed by the algorithm. The session will be closed at $e_i = \inf(t : Z(t) \geq B, t \geq s_i)$. In the language of stochastic processes, $e_1, e_2, \ldots$ are *stopping-times*, and the choice of B can be motivated by trying to control their statistical behaviour relative to the 'true' session end times. If the e_i tend to under-estimate then sessions are closed prematurely, producing a fractured picture of the network behaviour. If the e_i tend to overestimate then we risk merging sessions.

Naturally we must have $e_i \geq s_i + B$. Outside of this, the probabilistic behaviour of e_i is unfortunately non-trivial, even for simple models for the events within a session. As a result in further investigations we may need to rely on simulation.

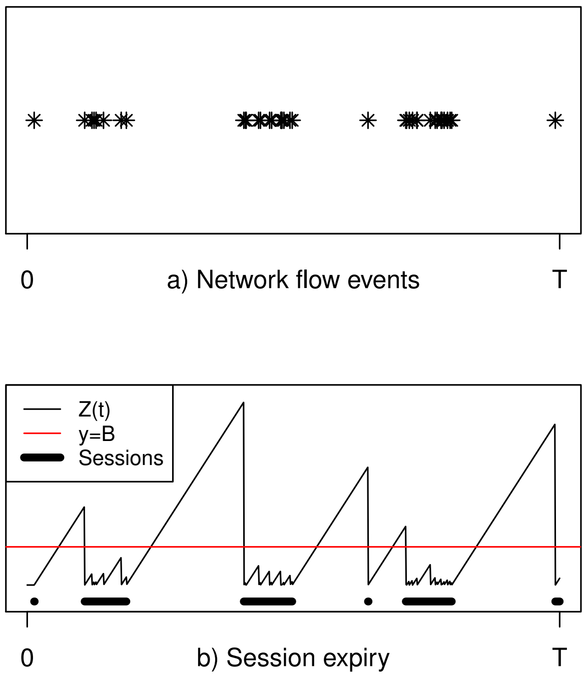

Fig. 1. Schematic of sessionization based on a time expiry policy. Figure a) shows simulated network flow events. Figure b) shows the time since the most recent event process, $Z(t)$, the chosen threshold B and the resulting sessionization of the data.

If it is possible to simulate a sample of event times from a given model then it is straightforward to simulate a set of boundary-crossing times $e_1, e_2, \ldots$, as in Figure 1b). Of course, a new challenge arises if there is uncertainty about the model generating the data within a session. In this case we may need an adaptive boundary $B(t)$, the height varying as we learn more about the process generating the session.

III. CLUSTER ANALYSIS

Instead of treating network flow events as a sequence of times, here we view the data as points on a one-dimensional space. From this perspective events being grouped into sessions can be restated as the points exhibiting *clustering* behaviour. The advantage of this insight is that it makes a large body of statistical literature on spatial point patterns relevant to the problem (even if they are normally motivated by 2 or 3D data).

Although it is known that network activity often occurs in sessions, in principle the signal might not be detectable in the data. The literature on spatial point processes helps answer this detectability question. Ripley's K-function, introduced by Ripley in two seminal papers [5] and [6], is a popular tool for measuring second-order dependence between points on a space. It can be used here as a preliminary check that there is 'session-like' behaviour in the data. To this end we compute

$$\hat{K}(d) = \sum k(x, y) T N^{-2},$$

where T is (as before) the length of the observation period, the summation is over all *ordered* pairs of points $x, y \in$

$\{t_1, \ldots, t_n\}, x \neq y$ that are within d of each other, and $k(x, y) = 2$ if y is more than x or $T - x$ of x, and 1 otherwise. $k(x, y)$ provides an edge correction for when x is close to the observation boundary (0 or T): it is inversely proportional to the chance that y would have been observed given its distance from x. (This correction is proposed by Ripley for a more general space: k is then the inverse of the proportion of the sphere circumference within the observation region.)

As a diagnostic for clustering behaviour the K-function can be computed and plotted for a range of values of d. Figure 2 shows $\hat{K}$ for the previously presented data (see Figure 1). This curve can be compared to that generated by completely spatially random (CSR) points, i.e. points that are uniformly scattered over $[0, T]$. Call this curve $\hat{K}_0$. For small d we should have $\hat{K}_0(d) \approx 2d$. To calibrate the observed difference between $\hat{K}$ and $\hat{K}_0$, a confidence band can be simulated around $\hat{K}_0$. Figure 2 shows a 95% confidence band (for each d, $\hat{K}_0(d)$ is within the band with probability 95%). $\hat{K}$ falling far above the band for small d is evidence of clustering behaviour. As a passing remark, $\hat{K}(d)$ passing below $\hat{K}_0(d)$ for small d would constitute evidence of self-inhibiting behaviour. We would expect to see this occasionally in flow data, because each flow is in fact not a point but a small interval of time.

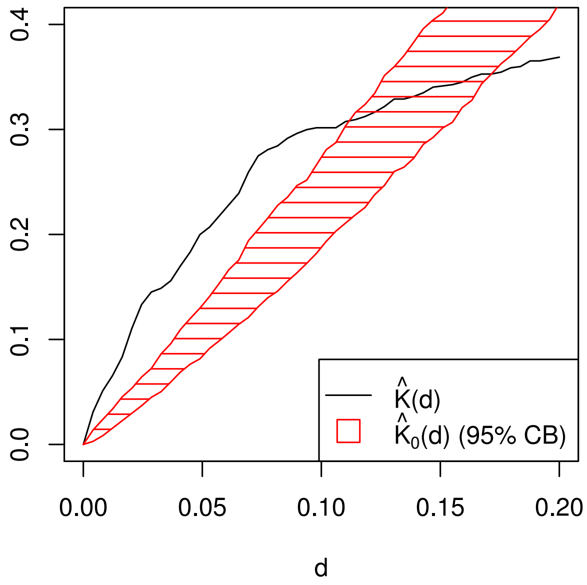

Fig. 2. The K-function. The black line shows the estimated curve $\hat{K}$, for the data presented in Figure 1. In red we show a 95% confidence band for $\hat{K}_0$ in case of CSR. $\hat{K}$ being largely above this band is evidence of 'session-like' behaviour.

The K-function also provides an inspiration for the following one-dimensional clustering algorithm (which is related to the 2D algorithm used in [7]). For a given range d compute $N_d(x) = \sum_{y \neq x} k(x, y)$, for each $x \in \{t_1, \ldots, t_n\}$. To standardise this quantity, note that for CSR we would expect $N_d(x)$ to be $2(n-1)d/T$ on average. To see this consider that we are constructing an interval of length $2d$ around each point, forming a region that is $2d/T$ smaller than the original for the $n-1$ remaining points to fall into. The edge correction makes this hold approximately even when the interval would otherwise extend outside the observation region. For each point we therefore compute $L_d(x) = TN(x)/[2(n-1)d]$ so that $L_d(x) > 1$ indicates that x is more clustered than expected under CSR. Then for a threshold τ assign all points $L_d(x) \geq \tau$ to a session, by making a link between any two that are not more than $2d$ apart and obtaining the resulting connected components, leaving the remaining unclustered.

Figure 3 illustrates this algorithm using $\tau = 1$, a reasonable default choice. Figure 3b) shows $L_d(x)$ for a given d, illustrated in the top-right corner. Figure 3c) shows the resulting sessionization of the points, with the open circles indicating which points are left unclustered.

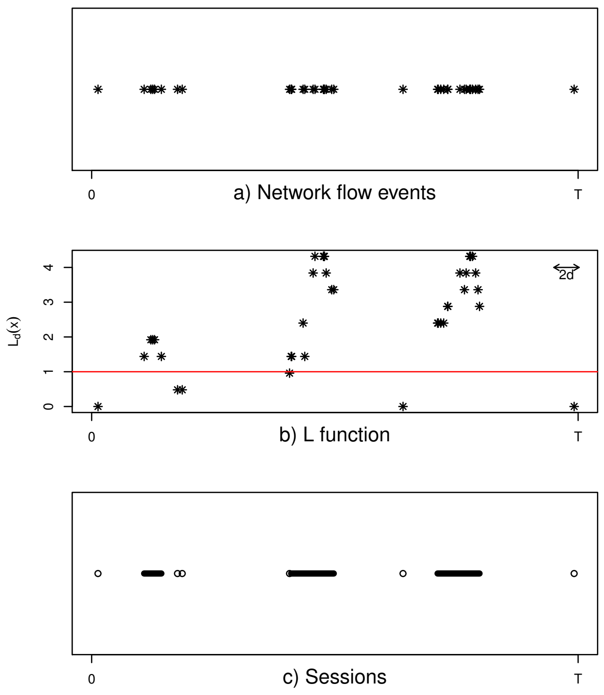

Fig. 3. Schematic of the L-clustering strategy. a) is a plot of the original data. In b) the black stars show $L_d(x)$ computed at each $x \in \{t_1, \ldots, t_n\}$ and the threshold $\tau = 1$ is shown in red. c) presents the resulting sessionization of the data, the open circles indicating which are left unassigned to a session.

We next investigate the choice of d and τ. From Figure 3 it is clear that the algorithm is sensitive to these settings since, for example, just a slightly smaller τ would have caused the second session to also contain the point directly to its left.

Consider that every combination of d and τ gives a proposed grouping of the events. Each proposal can be represented by a vector of labels $\ell = [\ell_1, \ldots, \ell_n]$ using the convention that unclustered points are assigned the special label 0, and all points sharing a non-zero label belong to the same cluster. In order to rank each proposal, we can devise a model for the data given ℓ, $p(t_1, \ldots, t_n|\ell)$, and formulate a Bayesian prior probability for each ℓ, $p(\ell)$. Then each proposal has a posterior probability

$$p(\ell|t_1, \ldots, t_n) \propto p(t_1, \ldots, t_n|\ell)p(\ell), \quad (1)$$

This can be used as a basis for choosing d and τ.

Let $c_0, \ldots, c_m$ denote the groups formed by ℓ. c_k contains the indices of the points contained in group k with c_0 containing the indices of the non-clustered points. Each cluster k

contains n_k points, which are denoted $t_1^{(k)}, \dots, t_{n_k}^{(k)}$. A simple example of a generative model for $t_1, \dots, t_n$ given ℓ is the following: for each group $c_k, k \geq 1$ draw the session start and end times by generating two independent Uniform random variables U_1, U_2 over $[0, T]$, (re)-labelled so that $U_1 \leq U_2$. Let $t_1^{(k)} = U_1$ and $t_{n_k}^{(k)} = U_2$. The remaining points of the cluster are then CSR over $[U_1, U_2]$. Non-clustered points are CSR over $[0, T]$. After some manipulation we find

$$p(t_1, \dots, t_n | \ell) = \frac{n_0!}{T^{n_0}} \prod_{k=1}^{m} \frac{2}{T^2} \frac{(n_k - 2)!}{\left[t_{n_k}^{(k)} - t_1^{(k)}\right]^{(n_k - 2)}}.$$

Combined with a prior on ℓ this formula can be used to optimise Equation (1) with respect to d and τ.

Note that in this perspective the L-clustering strategy is just a search procedure for optimising Equation (1). Other possible approaches to the problem include:

1) Agglomerative clustering [8]: starting with all points assigned to c_0, iteratively merge clusters until an optimum is found.
2) k-means [9]: for each m minimise
$$\sum_{k=1}^{m} \sum_{i=1}^{n_k} [t_i^{(k)} - \mu_k]^2,$$
where μ_k is the mean of cluster k. Then choose m to optimize Equation (1).
3) Kernel density clustering [10]: centre a kernel at each point with a given bandwidth w. The sum of the kernels is a multi-modal curve. Hill-climb from each point to the local mode, thus grouping together points that go to the same mode. Optimize Equation (1) with respect to w.

IV. A REGIME SWITCHING PROCESS

A regime switching process (RSP) is a time-evolving stochastic process where, at random points in time, the generative model for the data changes. Thus the process is said to switch between regimes, each regime being the model driving the data between two consecutive changepoints.

To obtain a partition of the timeline that can be interpreted as a sessionization we allow a special regime, denoted r_0, during which events occur at a very low or zero rate. Then the sessions inferred by the RSP are the periods between pairs of consecutive changepoints, excluding those driven by r_0.

This approach is both very general and very powerful, but it also presents various challenges. The first is statistical: how can we jointly learn the regimes and changepoints from the data? Naïve modelling strategies can give spurious results, for example, by placing pairs of changepoints tightly around every event. This happens to be the most compelling explanation for the data, that the rate of events was zero where there were no events and infinite at every event. As a result to obtain sensible results one must usually take a Bayesian approach or use a penalised likelihood.

The second challenge is computational. The Pruned Exact Linear Time (PELT) method [11] provides a fast solution for finding changepoints, assuming some regularity conditions on the objective function. This could be a Bayesian posterior, or a penalised likelihood, or some other measure of goodness-of-fit. As its name suggests the method can be as fast as linear in the number of events. On the other hand, if a Bayesian approach is taken then proper inference will typically require some form of stochastic approximation. For instance as part of a very influential research paper describing Markov Chain Monte Carlo on general probability spaces, [12] proposes a solution to the multiple changepoint problem when the rates between changepoints are IID. This implementation is possible here after changing the prior on the rates to be a bimodal distribution — a high mode for active periods and low for r_0. However to fit a more complicated model would require a specifically tailored algorithm, especially if we want to allow for repeated regimes and shared parameters.

V. CONCLUSION

We have presented three statistical perspectives on sessionization of network flow data, leading to a variety of algorithmic approaches to the problem. These differ in their computational complexity, what they assume about the data and whether they can be deployed in real time.

Throughout this article we have almost exclusively focussed on the start times of network flow data, ignoring all other information. Whilst this is motivated by a need for a simple and robust solution, there is clear scope for improvement by using the other information in the flows. However, our hope is that the sessions constructed by these relatively simple approaches are already enough to feed through to higher-level network analyses.

REFERENCES

[1] J. Neil, C. Hash, A. Brugh, M. Fisk, and C. Storlie, "Scan statistics for the online detection of locally anomalous subgraphs," *Technometrics*, vol. 55, no. 4, pp. 403–414, 2013.

[2] D. Lawson, P. Rubin-Delanchy, N. Heard, and N. Adams, "Statistical frameworks for detecting tunnelling in cyber defence using big data," School of Mathematics, University of Bristol, Tech. Rep., 2014.

[3] H. Zimmermann, "OSI reference model–the ISO model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 1980.

[4] E. Lehmann and J. Romano, *Testing statistical hypotheses*. Springer, 2006.

[5] B. Ripley, "The second-order analysis of stationary point processes," *Journal of applied probability*, pp. 255–266, 1976.

[6] ——, "Modelling spatial patterns," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 172–212, 1977.

[7] D. Williamson, D. Owen, J. Rossy, A. Magenau, M. Wehrmann, J. Gooding, and K. Gaus, "Pre-existing clusters of the adaptor lat do not participate in early T cell signaling events: supplementary material," *Nature immunology*, vol. 12, no. 7, pp. 655–662, 2011.

[8] S. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[9] S. Lloyd, "Least squares quantization in PCM," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[10] A. Hinneburg and H.-H. Gabriel, "Denclue 2.0: Fast clustering based on kernel density estimation," in *Advances in Intelligent Data Analysis VII*. Springer, 2007, pp. 70–80.

[11] R. Killick, P. Fearnhead, and I. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[12] P. Green, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika*, vol. 82, no. 4, pp. 711–732, 1995.