

# White Paper: Traffic Counting Methods

By Kevin Littlejohn for Obsidian Consulting Group

<http://www.obsidian.com.au/>

## Abstract:

As a first step to billing a customer (or being able to perform cost-recovery in a tertiary environment) for their usage of your network, collecting data about that usage is a requirement. Whether building a network from scratch, or adding billing to an existing network, some thought is required to be sure that the correct usage data is being recorded at the correct points, in the correct manner. This whitepaper attempts to compare and contrast some ways of collecting usage data in a network, with a view to giving the reader a better understanding of the options available.

Accounting and Authentication are tied together, often performed by the same device. In this whitepaper, we attempt to address accounting only, but will note where authentication issues might impact collection or collection issues might impact authentication.

This whitepaper also focusses on billing traffic usage over a network. There are many other things we have been asked to build billing for, from web usage through to photocopying (and, of course, voice services). We will touch on voice, but largely these other systems are out of scope for this document.

The accounting methods mentioned below have all been considered as accounting methods by users of Jet, Obsidian's billing solution. For more information on Jet, please see <http://jet.obsidian.com.au/>

## Note:

Throughout this paper we talk about "billing" and "customers" simply for the sake of convention. For environments where generation of customer revenue is not the primary driver for deploying a "billing" solution (e.g. tertiary institutions), it is valid to substitute "cost recovery" and "users".

## Generic traffic accounting issues

In any network, there are a few guiding principles to keep in mind when designing your traffic counting infrastructure.

First, decide where in the network you want to count traffic. As a rule of thumb, counting nearer the user will allow a tighter link between authentication and accounting, but be less scalable/require more resources as numbers of users scale up. In addition, many of the accounting methods will report usage "on the wire" - packet encapsulation and all. This wrapping will be different depending on packet size and type of media, and if you're not counting on your upstream link, your upstream provider is probably billing you for a slightly different amount of data (depending on where they count in their network, that may be the case regardless). Packet overheads alone can account for up to 20% of traffic count, so keep this in mind when you get to reconciling upstream reports with your own measurements.

Where you collect your usage data will also impact network design. Particularly if you have multiple provider links, or multiple sites, care has to be taken to count traffic only once (even in the presence of strange routing conditions such as links failing and traffic flowing through alternate routes), and to count all traffic. Often, you'll find that the provider-edge means of collecting data give you very little information about who did the traffic (and can completely miss traffic internal to the network or between network users), while customer-edge collection of data means lower-spec hardware doing the collecting, which leads to less differentiation and less overall information about the traffic.

While authentication and accounting are not directly linked, there are some areas of cross-over. Perhaps the most problematic is where a network manager wishes to disallow access to resources based on some form of quota or account balance. In the case of initial authentication, this means ensuring that your database of users and account balances is up-to-date and being referred to by the authentication system – usually not a huge problem. In the case of "prepaid" style accounts, where a user may run out of quota mid-session,

disconnecting an existing session can be extremely awkward. Most accounting systems do not allow for any form of kick-off process, and it's up to the billing system and access device to implement a scheme – usually this involves coding to a proprietary interface. This requirement can be worked around in some cases by carefully defining resources costs in terms that can then lead to (for example) timeout values on sessions – but this depends heavily on your billing policies and equipment in use.

Proxy servers can be a real problem for any traffic counting mechanism that does differentiation (eg. Netflow, SSG). These counting systems will give you the source and destination as the device sees them – that may mean that one side of the transaction (depending on location of proxy) ends up being the proxy server for most traffic. If that's the user's side, you've lost the ability to assign usage to them. Either move your proxy, or switch to transparent proxying (where the source IP is not changed by the proxy device). See "proxy logs" further down for some information on merging netflow (and other on-the-wire counting systems) and proxy data.

## A run-down of accounting methods

### SNMP

SNMP, or **S**imple **N**etwork **M**anagement **P**rotocol, is a very old protocol still in common use. It provides for queries and traps for monitoring and sometimes managing networked devices.

It is actually rare to see traffic counting via SNMP these days. Probably the most useful place for this is in a co-location facility, where you might have switches that are SNMP-capable and wish to track all traffic down each link as belonging to particular users.

SNMP will only give you bulk traffic figures – there is no differentiation of traffic types whatsoever. So if you want to bill all traffic, and not differentiate between national and international or in network versus external, then SNMP will fit. It is best suited to long-term or permanent links, as SNMP indexes for dynamic connections such as modems often change per connection (requiring more overhead for the accounting system to track who owns which counter).

SNMP provides no authentication mechanisms for your users, it simply reports traffic over a channel or link, and continually increases. It is up to the accounting system to allocate usage to the correct users and to deal with counter roll-overs or device reboots (which will reset the counters).

SNMP is available on pretty much any networked device in one form or another – right down to managed switches. In some cases, SNMP is a good option for adding kick-off facilities to a system that doesn't already have them (see RADIUS and others below).

### RADIUS

RADIUS (**R**emote **A**uthentication **D**ial **I**n **U**ser **S**ervices) provides both authentication and accounting for devices that connect to a network for a relatively short period of time – eg. Dialup modems, or wireless. Once a user is authenticated, RADIUS-capable devices can be configured to send interim accounting packets detailing how much traffic has been used to date in that session, and will send an accounting packet at the end of the session (when the user logs out or hangs up) giving how much traffic has been done across the whole session. As its name suggests, it was originally designed to fill a need for authentication for dialup NASes (Network Access Servers).

RADIUS's main advantages are as follows: It's a very simple, very widely spread and well-supported standard; it ties authentication and accounting together very strongly, and it's easily extensible. Anything that operates as a RADIUS server for authentication almost certainly also collects RADIUS accounting data and presents it in some useable form, be that databased or via logfiles.

The main disadvantage is that it's a very simple "bulk" figure – RADIUS will only ever give you the total traffic transferred over the link to the user, with no differentiation for different traffic types. If this is all you need, then RADIUS is well-suited to your needs.

The only other potential drawback is handling of interim packets. Some billing systems will not deal well with interim packets, and as a general rule for any "long-term" network connection (eg. Broadband/aDSL) RADIUS is not a great solution. RADIUS also doesn't offer any mechanism for disconnecting a user – see above "generic issues" for comment on this.

RADIUS is used as the basis for other mechanisms below – in particular, 802.1x and SSG. This is because it's a very flexible and fairly robust system. It is fairly common to see other systems relying on RADIUS as a carrier for their own data – see the captive portals below for other examples of this.

## 802.1x

802.1x is usually discussed as an authentication mechanism. However, when deploying an 802.1x based network, accounting is worth considering. While there is no actual hard definition of an accounting method for 802.1x (as it rightly describes a mechanism for forcing authentication at media level), it is expected that RADIUS will be used for authentication, and by extension accounting. RFC3580 discusses this use.

Because this relies on using RADIUS, it suffers all the same disadvantages as RADIUS – see above for more information. Probably the most problematic aspect of this is that there is no differentiation between traffic “on the network” (ie. Inside your own network), and traffic outside the network (which is the most likely differentiation for uncharged versus charged traffic). On the upside, 802.1x will give you everything the user does – there's no chance of missing any traffic to the user.

## Netflow

Netflow was initially a Cisco creation, however it has been taken up as a de-facto standard for collecting detailed accounting information, and a number of other vendors have added some form of Netflow support to their own devices.

Netflow apparently originated as a debug aid for the (then) new generation of routers, that did traffic routing based on flow switching – instead of calculating best path for every packet as it came through the router, a “flow” would be defined, made up of source and destination IP and port and protocol, and the routing decision made once at the start of the flow. Netflow provided a way to extract information about how the router was making those routing decisions in real-time – essentially, it exports the router's internal flow-data tables in (by default) 15 minute slices.

This means that Netflow provides an excellent level of detail – from it, you can gather source and destination IP and port, protocol, various ToS/QoS flags – pretty much any detail that might have had routing implications. From this, you are free to aggregate in whatever way you choose prior to billing.

The main advantage of Netflow is that it's so detailed. This also poses an interesting challenge, however – particularly when scaling up to large networks, the act of aggregating can become fairly involved and resource-intensive in itself. On larger networks (eg. Gigabit or above pipes), use of Netflow is problematic (latest versions of Netflow actually move away from full information toward a sampling-based solution, which is fine for network analysis but possibly not appropriate for billing).

Another potential issue with Netflow is lining up authentication and accounting. Netflow has no authentication component – so tracking exactly who owned the IP number when traffic was recorded can be a problem in some environments. Worst-case scenario is shared access computers, where people login and logout of the network, but Netflow is simply counting traffic at the border. Netflow will cache information about existing flows for up to 15 minutes (configurable), so two people using the same IP within a 15 minute segment may mean mis-assigned data.

Netflow is also usually only available on larger routers – small routers and edge NASes (Network Access Servers) often don't provide any form of Netflow export. This means that a Netflow solution usually involves collecting traffic counts near the provider edge of the network, or in the core – see the “generic issues” section above for the implications of this.

Also, while Netflow does tag which interfaces were incoming and outgoing, this may not be enough (depending on network layout) to determine direction of traffic. A fully-fledged Netflow system for a large tier-1 or tier-2 style network (with suppliers and consumers spread around the network edges) will involve some fairly intricate aggregation and assigning code. Luckily, for the vast majority of networks, this is not an issue – it's often fairly clear which IPs are internal and which are external.

The most sophisticated application of Netflow data this author has seen/worked on involved using Netflow to collect traffic counts, then matching in real-time with BGP table information to determine via AS-path whether the traffic was national or international (with some small degree of error). BGP path information was also used to determine whether the traffic was destined for an end user directly, or whether it was being routed via another end user (customers with links to the ISP in question and to reseller's of the ISP,

where their traffic changed route depending on reliability of link and/or cost of traffic). This had to happen in real-time, to ensure correct BGP data, then the traffic was further aggregated later to give a bulk figure per slice rather than a total per-flow breakdown. The important thing about building a system this involved is to make sure each step is independently repeatable – so if one link in the chain fails, the prior portions of the system continue to collect data that can be back-processed later, and so that if a bug is found in one step late in the day, the data affected can be re-processed to correct the error. An interesting side effect of this approach was that per-pipe differences in charging were no longer possible, as the Netflow system could identify which user the traffic was destined to, but not via which pipe. These sorts of considerations can have an impact right up to the actual plans your organisation can offer due to network design.

## **SSG**

SSG can be considered the “next step up” from Netflow for Cisco devices. It provides both authentication and accounting, and leans very heavily on RADIUS. To give some idea of how SSG operates in authenticating a user, we present the layout we use with Jet + SSG:

- User attempts to browse the Internet via web browser, SSG-capable router intercepts traffic and redirects user to a captive portal login page provided by Jet.
- Jet prompts for and collects username and password. Jet then sends a RADIUS login query to the SSG-capable router.
- SSG router sends a RADIUS login request (a modified version of the one we sent) to Jet.
- Jet checks it's database, ensures user has current account balance and is allowed access to the services requested, returns a RADIUS auth success with appropriate services marked (or a reject).
- SSG router returns a success message (or reject) to Jet, and adds client's IP to it's own list of allowed clients.
- Jet redirects user's web request to the initial page requested.

SSG adds some vendor-specific attributes to the standard RADIUS packets (this is allowed as part of the RADIUS specification, see section 5.26 of RFC2138). These attributes are not well documented, but are also not hugely difficult to work out – they're essentially character strings to indicate which services should be allowed or disallowed. Services are setup akin to access lists on the SSG-capable router.

Because SSG uses RADIUS but differentiates based on services allowed, when the user logs out the billing system will get multiple accounting packets (this applies to Interim packets also) – one for each service, and one overarching “totals” packet. This can be used to provide differentiation similar to Netflow, with the advantage that the router itself aggregates before sending the data. Because this is RADIUS-based, the traffic is strongly matched to the user responsible – no chance of mis-assignment. In addition, Cisco have provided a mechanism to “kick off” a user, by sending a specially-crafted RADIUS packet to the router. This allows a billing system to, on receiving an interim accounting packet, determine the user is out of quota and disable their access – effectively pushing their web browsing back to the login screen, where they can be prompted to purchase additional quota.

The SSG approach suffers only a couple of small flaws. Firstly, being a captive portal, it requires the user to login via a web browser before getting any access to the rest of the network – this usually turns out to be acceptable. Secondly, because of the processing requirements, SSG is only really available on higher-end Cisco routers. These two combine to encourage placing SSG choke points at the provider edge of the network – which works very well for capturing data use from your upstream to your client, but can mean you miss counting traffic use inside the network or between your users, where it does not traverse your external links.

## **Captive Portals – NoCatAuth, WifiDog**

There are a number of captive portal systems available today – the two mentioned above are OSS (Open Source Software) solutions. These have sprung up around wireless networks, and the need to ensure users authenticate before gaining access to the wider network.

In most cases, these systems have either built their own databases for users, or have leaned on RADIUS. They offer a “captive portal” web approach to logging in – the end user is redirected on first web browse to

a web page allowing them to login. In the case of accounting, the two above have been singled out because they provide accounting data via RADIUS.

While these systems have been initially designed for wireless networks, they can be re-used for fixed-line networks (eg. office computers or buildings) – they simply ensure that the user logs in prior to using the network beyond themselves, and provide accounting data.

The only real caveat with these devices is the previously-mentioned issues around kicking users off mid-session – each device in this sphere offers either nothing, or a custom system for an external device to kick a user off. Luckily, because these packages are OSS, it's not impossible to extend them in the required ways.

## **Network Management Devices**

These devices have been grouped together, as they are substantially similar in their approach to the network and accounting. We have implemented authentication and traffic accounting for a number of network management devices -- all of whose vendors are exhibiting at QUESTnet, so we suggest you seek them out and talk to them about their products.

When integrating with such devices, we work in the following way:

- User attempts to browse the Internet via web browser, device intercepts traffic and redirects user to a captive portal login page provided by Jet.
- Jet prompts for and collects username and password. Jet then checks to see if user is allowed to login
- If login allowed, Jet contacts the device via the appropriate vendor API and re-configures it's firewall rules such that the user's IP is allowed access to the services indicated in the Jet database.

The advantages these devices provide are mostly in terms of the other services they supply. They can be a very powerful tool for network analysis and control – talking direct to them from their own clients, an administrator can get a lot of detail about what's going on in the network, and reach in to block various different types of traffic in real-time.

One other really large advantage is that these devices are capable of applying rate limiting or traffic shaping to individual users – if properly configured and managed by the accounting system, they can provide unparalleled ability to control the shape of your user's usage.

The main disadvantages these devices present to someone implementing an accounting solution are that, as they are a fairly new class of device, they often have their own proprietary interface, and are designed to support direct manual control of users and traffic very well, sometimes to the detriment of automated accounting. They all (so far) present accounting data in a batched form, requiring the accounting system to poll to retrieve data, and because they're very generic devices they don't really have a full concept of a logged-in user (so they can't report user logout/end of session and traffic associated ala a la RADIUS). This leads to extra work to poll to get data, and to implement a particular configuration per site for enabling/disabling access to users.

## **Proxy logs and other application-level logging**

In some cases, your network will have some obvious choke points around proxy servers or other application-level devices (eg. Socks servers). These devices can provide authentication and access control, and also some logging of data for accounting.

As there is a huge range of such devices, there is also a huge range of ways the logs can be formatted. When considering accounting from a logfile, you should look for the presence of the following fields: start time, end time, username (or other user identifier), amount of traffic, type of traffic. If any of these are missing, then there'll be extra work required to determine these as part of post-import processing of the logfile. In particular, if the username is missing, you may be buying into cross-referencing between your accounting system and your authentication system ("who owned this IP at that time?"), which is doable, but has to be designed with care.

Bear in mind, also, that application-level logs will give you actual dataload traffic figures, not data + packet headers – your provider is almost certainly charging you for on-the-wire overheads also. This will lead to a disparity (and, if you start trying to calculate left-over amounts by subtracting proxy logs from gateway netflow, for instance, you'll be introducing inaccuracy).

## Vendor-provided logs

If re-billing an upstream service (eg. ADSL in Australia), the actual provider of the physical and IP-level service may provide you with daily logs of usage. In this case, it's a matter of treating these similar to proxy and other application logs as above. Building a set of processes to collect, read, and input that data is not usually a large operation, but be aware of the amount of detail you get in those logs – it will be a significant factor in determining how complex your plans can be.

Also, note there is a time delay in retrieving these log files, which means that prepaid usage is probably a bad idea – you may find a delay of up to a day between the user going over their allowance, and you getting the logfile saying so.

## Voice/VoIP Services – Asterisk and similar PABX systems

Voice over IP is an emerging market, and of great interest to a wide range of people right now. We present one option for collecting usage data – Asterisk – as this is the most common Open Source application used for controlling VoIP (and other devices can reasonably be presumed to provide similar sorts of access to data).

Usage data for voice is subtly different to usage data for IP traffic. In both cases, there is a charged party and a non-charged party (source and destination), in both cases there is an amount of traffic done. In the case of IP traffic, there may be differentiation based on the external IP – in the case of voice, this translates to different call charge rates for different categories of B-end (national, international, mobile, etc).

In the case of voice, there is usually only a single record per call – no interim packets. Usage is charged on time rather than data. The interesting issues usually come in later in the billing process – in the way usage is charged (for instance, bands of usage within a single session (first 15seconds free), flagfall charges, bundling of charges (free national calls if international are over \$20 worth), and similar).

Asterisk is highly flexible when it comes to logging accounting data. It can log to file or straight into a MySQL database (logging to file is arguably more reliable, but also involves a little more processing and can introduce a time lag between usage and billing). Fields can be added or removed to the logs via configuration options, and the dial plan can be used to further categorise traffic.

Ideally, Asterisk is configured to log each line with two pieces of information: The username or other identifier for the user who should be charged for the call, and the category of traffic for rating (national vs. international vs. mobile vs. any other charging categories). If those two pieces of information are available in the logs, then all the logic of working out what sort of traffic it is has taken place in Asterisk, and the billing system that works with the accounting data can be concerned only with how to turn that accounting data into a dollar charge. If either piece of data is missing, then either your accounting system, or a separate piece of software between Asterisk and the accounting system, will need some logic to determine either which user owned the call (based on A and B ends), or what type of traffic it is (based, usually, on lookup tables for phone number prefixes, ideally done in the dial plan, as close to the routing decisions as possible).

One small gotcha with voice-call logs – they, like Netflow, work in terms of "A end" and "B end" (in Netflow's case, that's source and destination). These are not internal and external addresses – so the first thing you have to do is work out which end of the call belongs to a chargeable client and which end belongs to an outside party (or, in the case of internal calls, who to bill the call to out of two clients). Unlike Netflow, at least the Asterisk logs give you an idea of who initiated the connection in the first place, and you only get one record per call/connection.

## Summary

This document has laid out a number of different ways to collect data about network usage. This is not necessarily exhaustive, and new systems are coming onto the market all the time. However, this does introduce some of the issues we deal with in every Jet implementation, particularly with larger networks. In any rollout, there are compromises to be made between network layout, desired user plans and quota schemes, and hardware costs – knowing some of the options can help you understand how to address these challenges.

At Obsidian, we specialise in helping customers deal with these sorts of issues of planning and integration. We can bring our experience with various network layouts, and particularly with common growth patterns of networks, to bear on your initial network designs and/or upgrade plans, as part of a Jet software rollout, providing you not only with top-class software for managing quotas and billing, but also a solid foundation for building an appropriate network, both now and for the future.