



pervasivet<sup>technology</sup>labs  
AT INDIANA UNIVERSITY

*www.pervasivet<sup>technology</sup>labs.iu.edu*

# Netflow - what is it, and why do we hate it?

David A. J. Ripley

Advanced Network Management Laboratory

Pervasive Technology Labs

Indiana University



pervasivetechlabs  
AT INDIANA UNIVERSITY

*www.pervasivetechlabs.iu.edu*

**David A. J. Ripley**  
MSc., ARCS

[daripley@indiana.edu](mailto:daripley@indiana.edu)

Lead Network Security Developer,  
Advanced Network Management Laboratory  
Indiana University

Network security infrastructure development and  
research for the ANML.

Background in physics, image processing, satellite  
remote sensing, system administration.



pervasivetechlabs  
AT INDIANA UNIVERSITY

[www.pervasivetechlabs.iu.edu](http://www.pervasivetechlabs.iu.edu)

# Overview

- What is a “flow”?
- What is Netflow specifically?
- Netflow collection infrastructure.
- Netflow processing, online and offline.
- Facts and figures, problems and issues.



# Netflow Recap

- Q. What is a flow?
- A. In a general sense, a flow is a series of packets with some attribute(s) in common.



# Netflow Recap

- Common attributes define a flow
  - Source and/or destination of the traffic.
  - Protocol - TCP, UDP, ICMP?
  - Timing - start, end, and duration of the traffic.
  - Routing information - interfaces, AS, etc.



# Netflow Recap

- Flows can be unidirectional or bidirectional - the latter adds possible information.
- Aggregated flows.
- Application flows - classify packets by inspecting their contents
- We're not going to worry too much about these cases.



# Netflow Recap

- As far as we're concerned, a flow is a series of packets with the same:
  - Protocol (UDP, TCP, ICMP)
  - Source and destination ports
  - Source and destination addresses



# Netflow Recap

- The recording of a flow is subject to idiosyncrasies of sampling frequency and sampling window
  - Bucket timeout - systems typically consider one minute windows.
    - Flows longer than one minute will appear as two flow records
    - Multiple flows (with the same characteristics) within a single one minute window will appear as a single flow record
  - Sampling rate
    - Router will only consider one out of every N packets;  $N=???$  - data loss vs. expensive operations.



# An example

- Host A gets a web page from Host B
- This will show up as **two** flows (usually)
  - Host A, port 12345  $\Rightarrow$  Host B, port 80
  - Host B, port 80  $\Rightarrow$  Host A, port 12345



# Why Netflow?

- What kinds of information can we gather?
  - What percentage of traffic on the network is web traffic? ssh? IRC?
  - What is the average transfer rate for network communications?
  - Who uses the network the most?
  - Have usage patterns changed over time?
  - For the Chicago region, how much of the traffic of the region is staying in the region?
  - Many others



# Why Netflow?

- **Historically, traffic accounting, acceptable use enforcement;**
  - Researchers and engineers needed to answer all kinds of questions about network traffic.
  - Traffic accounting in the form of flow records provided that information.



# Why Netflow?

- **Traffic Engineering/Accounting**
  - How traffic is shared with competitors; how customers are billed.
- **Security/Policy monitoring**
  - DoS/DDoS detection
- **Research**
  - Measuring the growth of networks
  - Identifying how the network is being used.



# What data is there?

- It depends.
- We keep talking about “flows” - we really mean Cisco’s Version 5 flow records
  - A Cisco-defined “standard”
  - Used on Abilene - so that’s what we use.



# Netflow Version 5

- Cisco-defined de-facto standard
  - Efforts are underway in the IETF to make this standard official
- Flows are exported as UDP packets
  - Each packet contains a number of flow records plus a header with information common to these records
  - Delivery is not guaranteed!
    - There are sequence numbers so we know how many packets we've lost.



# Netflow V5 Header

Byte 1	Byte 2	Byte 3	Byte 4
Version		Count	
SysUpTime			
UNIX Seconds (seconds since Epoch)			
UNIX Nanoseconds (residual nanoseconds)			
Flow Sequence Number			
Engine Type	Engine ID	Reserved	



# Netflow V5 Record

Byte 1	Byte 2	Byte 3	Byte 4
Source IP Address			
Destination IP Address			
Next Hop IP Address			
Input ifIndex		Output ifIndex	
Packets			
Bytes			
Start time of flow			
End time of flow			
Source port		Destination port	
Padding	TCP Flags	IP Protocol	TOS
Source AS		Destination AS	
Source Mask Length	Destination Mask Length	Padding	



# V5 Header Details

<b>Version</b>	In our case, 5
<b>Count</b>	Number of records in packet
<b>sysUpTime</b>	Milliseconds since boot
<b>Unix Seconds</b>	Number of seconds since epoch
<b>Unix nanoseconds</b>	Residual nanoseconds
<b>Flow sequence number</b>	Sequence number for flow packet
<b>Engine Type</b>	User defined
<b>Engine ID</b>	User defined



# V5 Record Details

<b>Source IP</b>	Source IP Address of flow packets
<b>Destination IP</b>	Destination IP Address of flow packets
<b>Next Hop IP</b>	The next hop to which the flow was forwarded
<b>Input ifIndex</b>	SNMP Index value for incoming interface
<b>Output ifIndex</b>	SNMP Index value for outgoing interface
<b>Packets</b>	Number of packets in flow
<b>Bytes</b>	Number of bytes in flow
<b>Start time</b>	Time (since boot) of flow start
<b>End time</b>	Time (since boot) of flow end



# V5 Record Details Cont.

<b>Source Port</b>	IP Source port
<b>Destination Port</b>	IP Destination port
<b>TCP Flags</b>	Logical <i>or</i> of all TCP flags seen
<b>IP Protocol</b>	IP Protocol value
<b>TOS</b>	Type of service bits set
<b>Source AS</b>	Source Autonomous System
<b>Destination AS</b>	Destination Autonomous System
<b>Source Mask</b>	Source address prefix mask bits
<b>Destination Mask</b>	Destination address mask bits



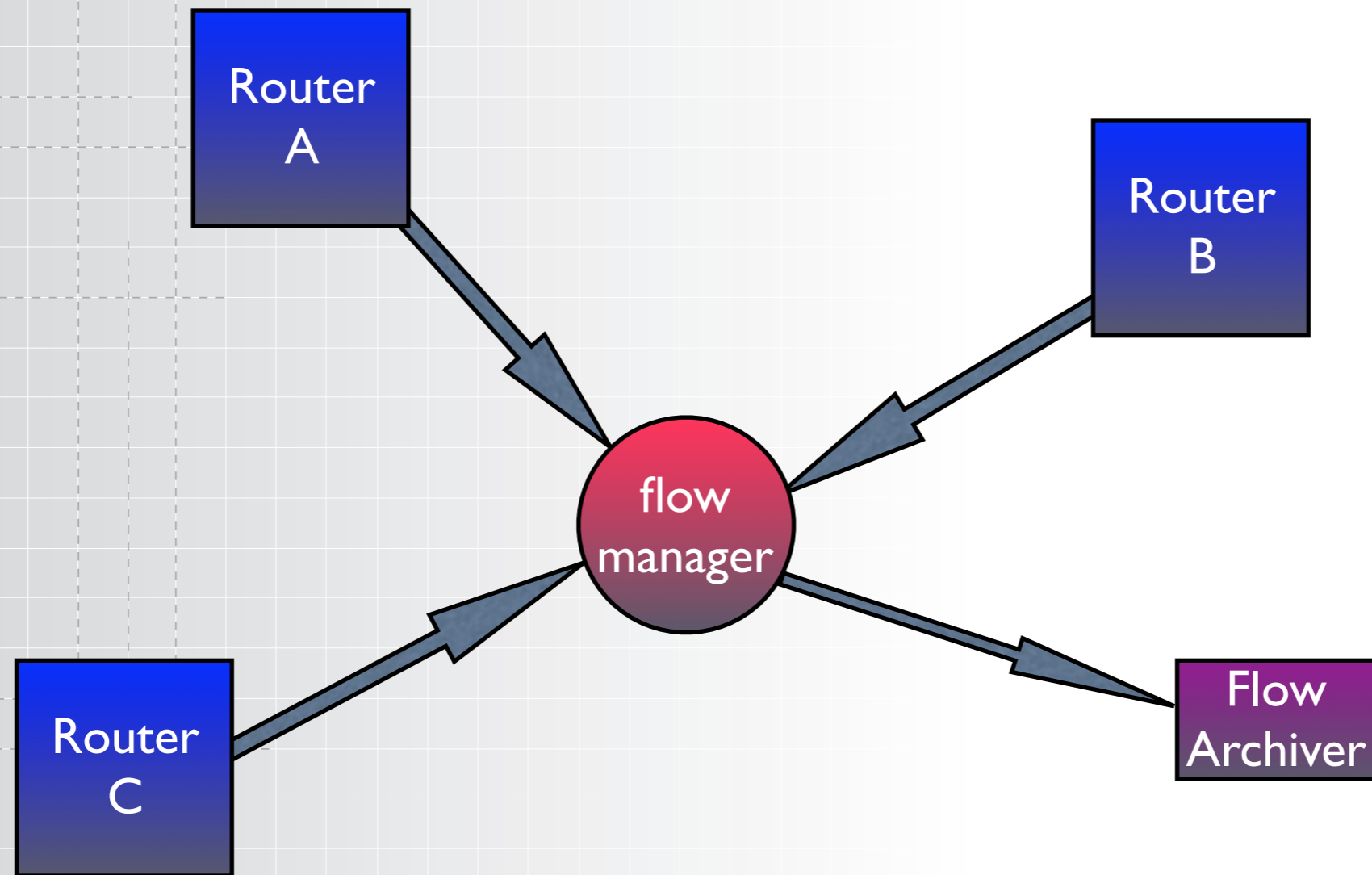
# Convenience, or lack of it

- Flow records are exported in a format that is convenient for the router, not for us.
    - e.g. The flow start and end times are in a form that is not immediately useful, milliseconds since system boot.
    - We have to combine data from individual flow records with header data.
      - Seconds since epoch is the Right Thing
- Flow StartTime = Unix Seconds + Unix Nanoseconds - sysUpTime + flow\_start
- (After we've converted all these to the right units)
- ICMP Type is stored in the destination port field



# Netflow Collection

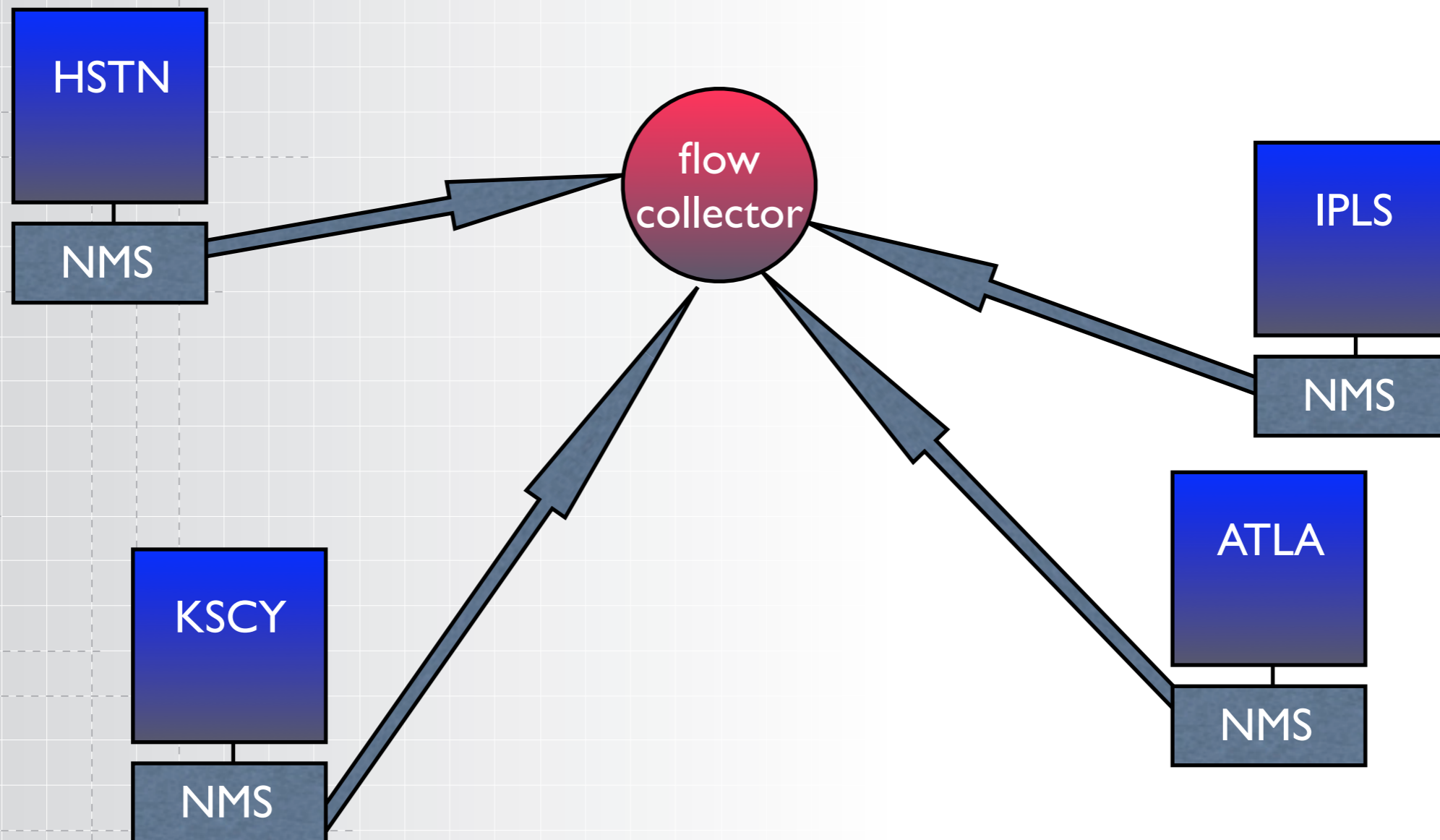
- A simple flow collection architecture





# Netflow Collection

- Closer to the truth...





# Netflow Collection

- ANML Flow collector is a Dual Xeon with quite a bit of memory and disk space.
- Collects flow data from Abilene core routers.
  - Archives raw records (up to 3 months)
  - Redirects to other lab machines



# Netflow Collection

- Much of the grunt work is done using flow-tools
  - (<http://www.splintered.net/sw/flow-tools/>)
- Also some gobs of glue (cron, perl)
- These take care of basic collection, archiving, expiration.
- flow-tools jolly useful for collection, filtering etc.
- Here's how we collect our data:

```
flow-capture -w /huge/flow -z9 -V5 -m 255.255.248.0 -E800G 0/0/4000 -S5
```



# Pump up the volume

- How much data gets collected?
  - We're looking at all of Abilene. Large, busy, network.
  - 3700-6000 flow records per second
  - ~424 million records per day
  - Almost 13 billion records per month
  - Even zlib compressed, this comes out at typically 200GB of data per month.
  - A lot!



# Examining Netflow

- Part of our job is using netflow data to see what's happened/is happening on the network
- We spend a significant amount of time processing the archived data looking for particular behaviors.



# Examining Netflow Data

```
$ flow-cat ./data | flow-nfilter -f filters -F tcp_only | flow-stat -f9 -S1
```

```
# --- ---- Report Information --- ----
```

```
#
```

```
# Fields: Total
```

```
# Symbols: Disabled
```

```
# Sorting: Descending Field 1
```

```
# Name: Source IP
```

```
#
```

```
# Args: flow-stat -f9 -S1
```

```
#
```

```
#
```

```
# IPaddr flows octets packets
```

```
#
```

```
130.49.72.0 30415 6881695 36110
```

```
128.223.216.0 28716 540202269 427566
```

```
130.49.88.0 24267 10770631 32186
```

```
etc.
```



# Examining Netflow Data

- Our example:

```
$ flow-cat ./data | flow-nfilter -f filters -F tcp_only | flow-stat -f9 -S1
```

- We have to have config files like this:

```
filter-primitive tcp_only  
  type ip-protocol  
  permit 6
```

```
filter-primitive udp_only  
  type ip-protocol  
  permit 17
```

```
filter-definition tcp_only  
  match ip-protocol tcp_only
```

```
filter-definition udp_only  
  match ip-protocol udp_only
```

- We'd like an easier way



# Fun with Perl

- Sometimes limitations of existing tools require rolling your own.

```
#!/usr/bin/perl -w
use Socket;
$i=<STDIN>;
while(defined($i=<STDIN>)){
    @fields = split /,/, $i ;
    $fields[10]=inet_ntoa(pack(N,unpack(N,inet_aton($fields[10])) &
0xfffff800));
    $fields[11]=inet_ntoa(pack(N,unpack(N,inet_aton($fields[11])) &
0xfffff800));
    print(join(", ",@fields));
}
```



# Might there be a better way?

- Looking for a more flexible way of querying data.
- But we didn't want to write our own tools.
  - Laziness, impatience (and hubris?)
- Might there be some Structured way to Query the data, perhaps even an entire Language?



# SQL? WTF?

- Why **not** put everything into a relational database?
  - There are plenty of reasons **not** to;
  - It's a lot of data.
  - There are existing ways to query the data.



# SQL? WTF?

- Why do it?
  - It makes it easy to construct very complex queries
  - It's (potentially) fast and efficient
  - Well-established interfaces with familiar front-ends
    - via ODBC etc.



pervasivetechlabs  
AT INDIANA UNIVERSITY

*www.pervasivetechlabs.iu.edu*

# Netflow database

- Set up a PostgreSQL Server
- Created a table or two
- Started stuffing data into it.



# Netflow Database

- Each row has all data from flow record, plus header data from packet.
- We didn't do *anything* to the data before dumping it into the table.
- Added some other tables (AS map)

```
netflow=# \d v5_rawnetflow_current
Table "public.v5_rawnetflow_current"
  Column      |  Type   | Modifiers
-----+-----+-----
 unix_secs    | integer |
 unix_nsecs   | integer |
 sysuptime    | bigint  |
 exaddr       | inet    |
 dpkts        | integer |
 doctets      | integer |
 first        | bigint  |
 last         | bigint  |
 engine_type  | integer |
 engine_id    | integer |
 srcaddr      | inet    |
 dstaddr      | inet    |
 nexthop      | inet    |
 input        | integer |
 output       | integer |
 srcport      | integer |
 dstport      | integer |
 prot         | integer |
 tos          | integer |
 tcp_flags    | integer |
 src_mask     | integer |
 dst_mask     | integer |
 src_as       | integer |
 dst_as       | integer |
```



# Lesson #1

- We learned very quickly that we were stepping over some kind of line
  - Shared memory (kernel)
  - Disk flushing
  - Numbers of buffers, size of buffers



# MS RPC Vulnerability

- First test!
- Look for machines showing signature behaviour;
  - We have no access to the contents of packets;
  - Host A sends to RPC listener on B; remote shell opens on B; A sends tftp GET command to remote shell; B gets executable from tftp server on A.
  - Exactly the sort of thing we wanted.



# MS RPC Vulnerability

- Broke this down into several tasks;
  - Server-side, digest the main archive looking for traffic on the ports/protocols we were interested in.
  - Query reduced data set looking for matching hosts (hit by scanning source, subsequent tftp activity)
    - We know that there are issues with this approach
  - (Bonus: AS to AS Name lookup.)

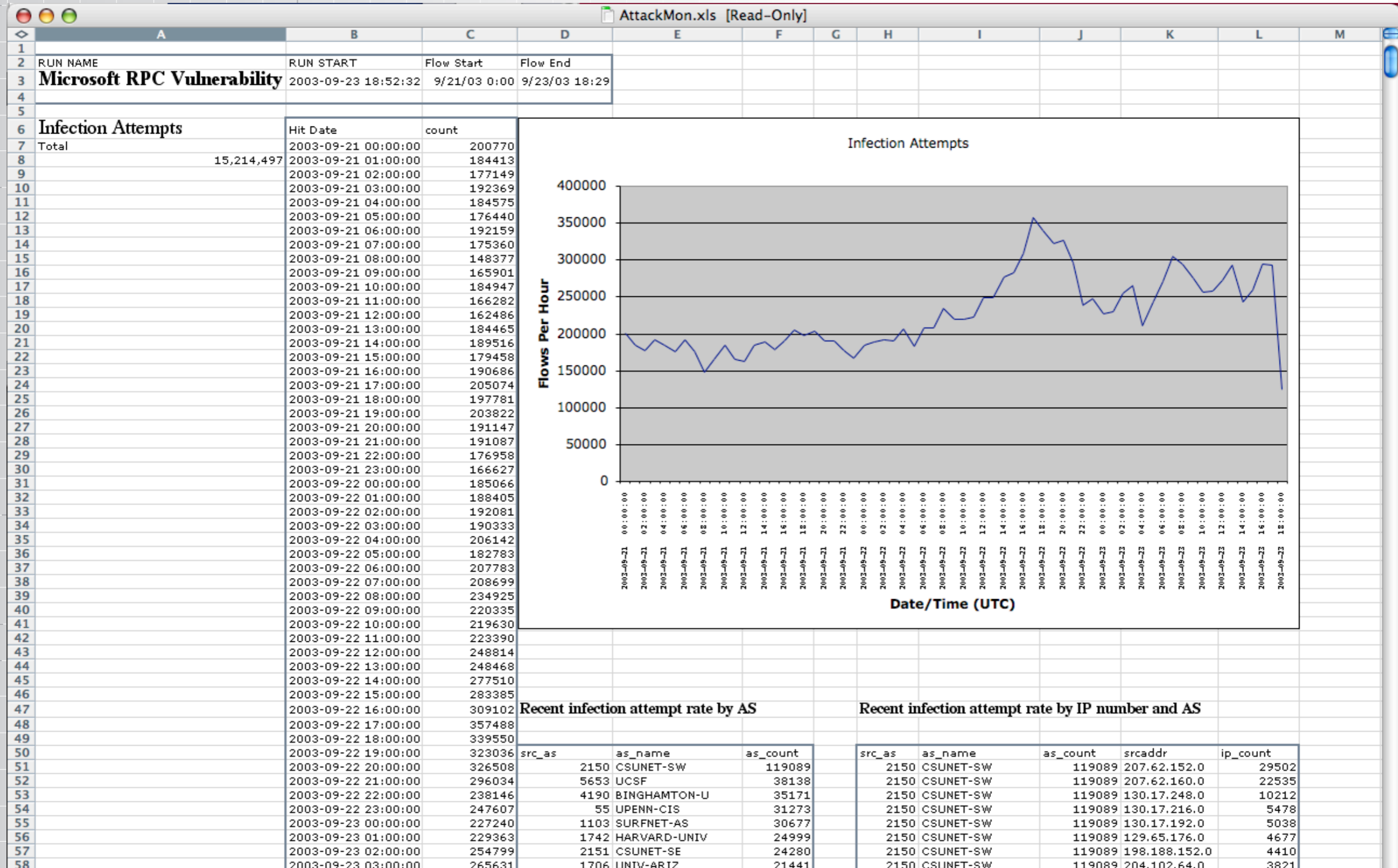


# MS-RPC Vulnerability

- The initial digestion of the raw flow records proved by far the most time consuming operation
- Subsequent operations were quite quick.
- Used ODBC to get this into a spreadsheet.

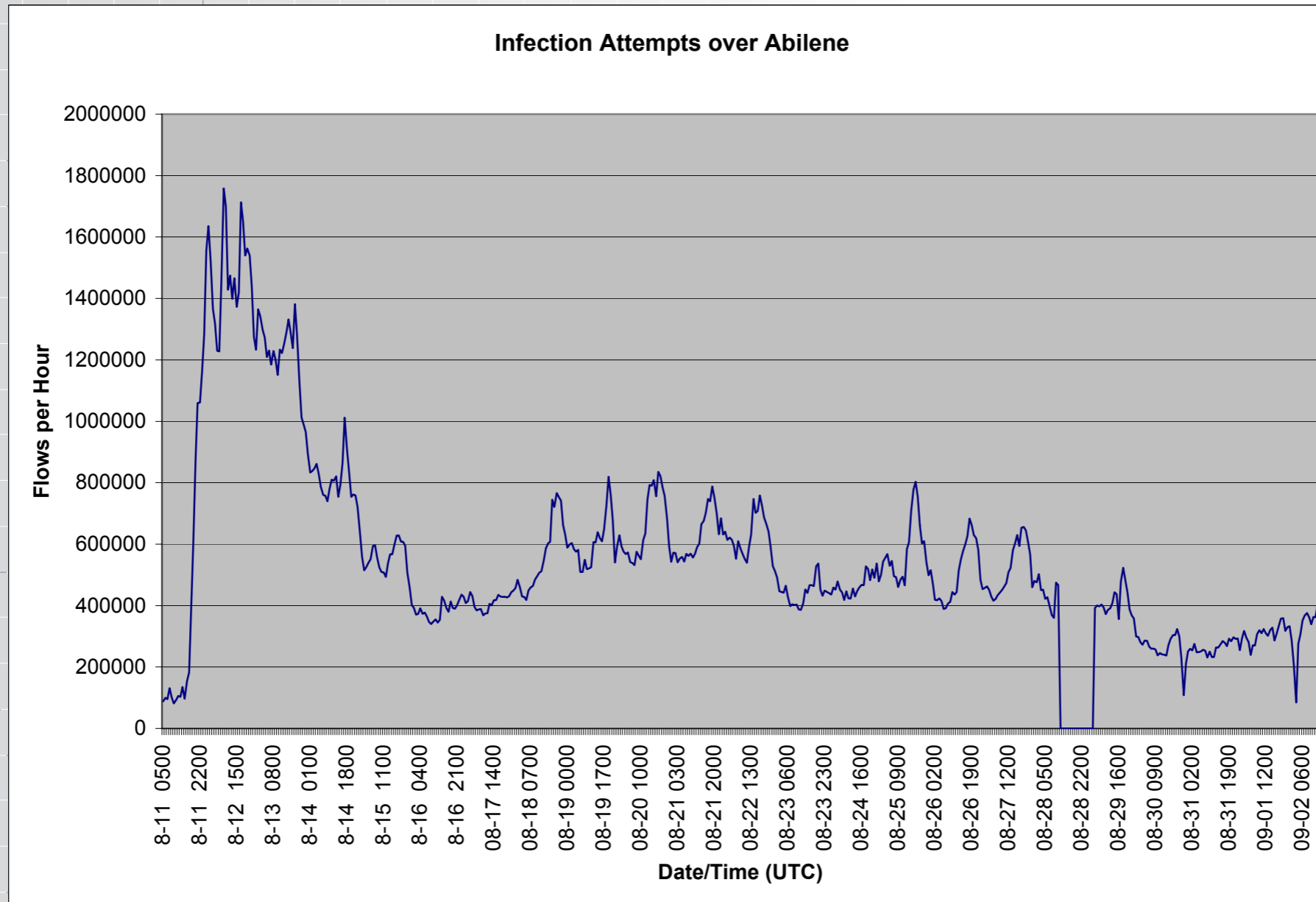


# MS-RPC Vulnerability



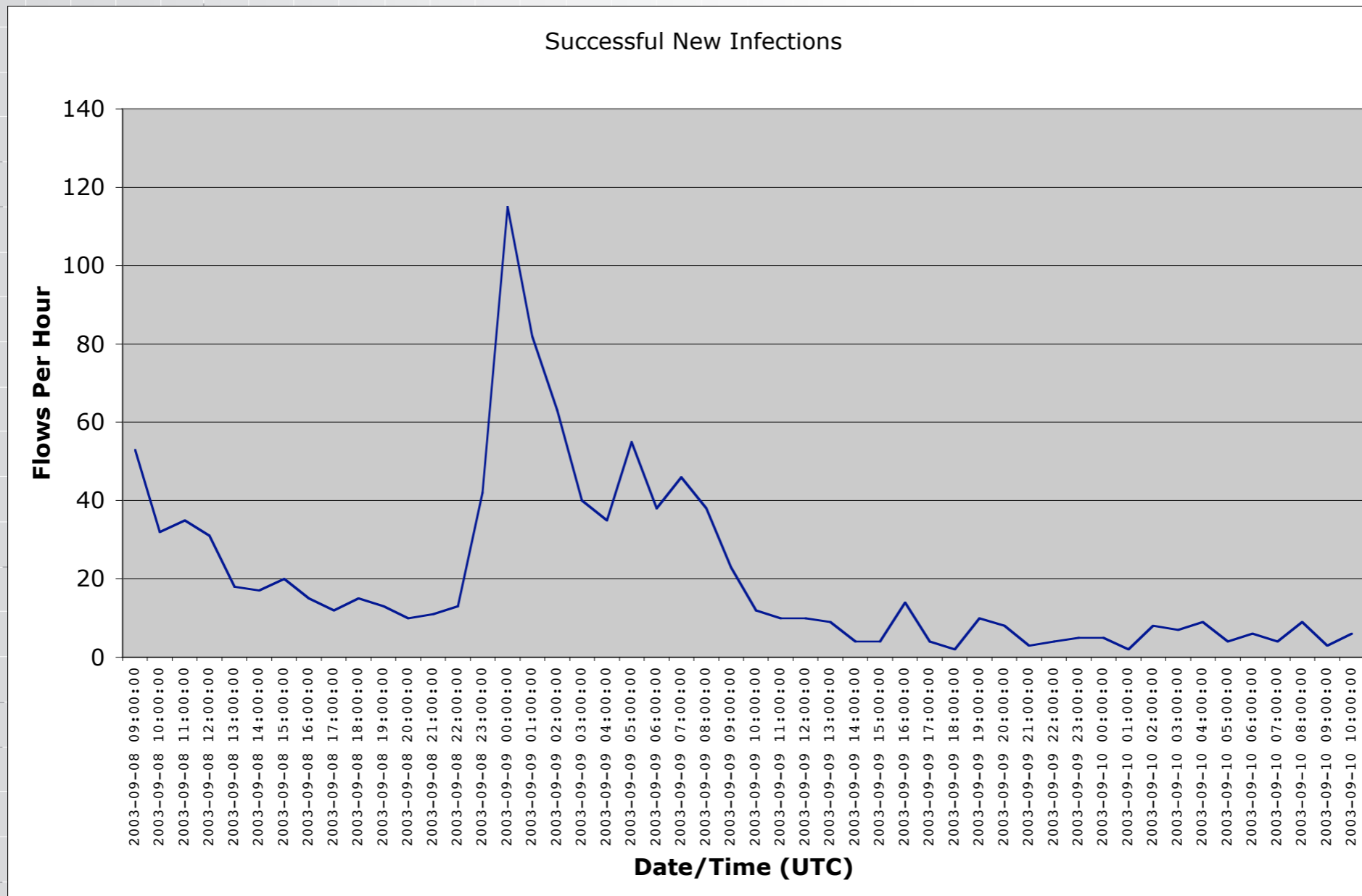


# MS-RPC (Attempts)





# MS-RPC Infections (Maybe)





# Added Value

- Some operations are very computationally cheap and easy to do as we go along, so it would be silly not to do them.
- Traffic to and from different ports, broken down by time.
- This lets us get basic information quickly and easily, e.g.

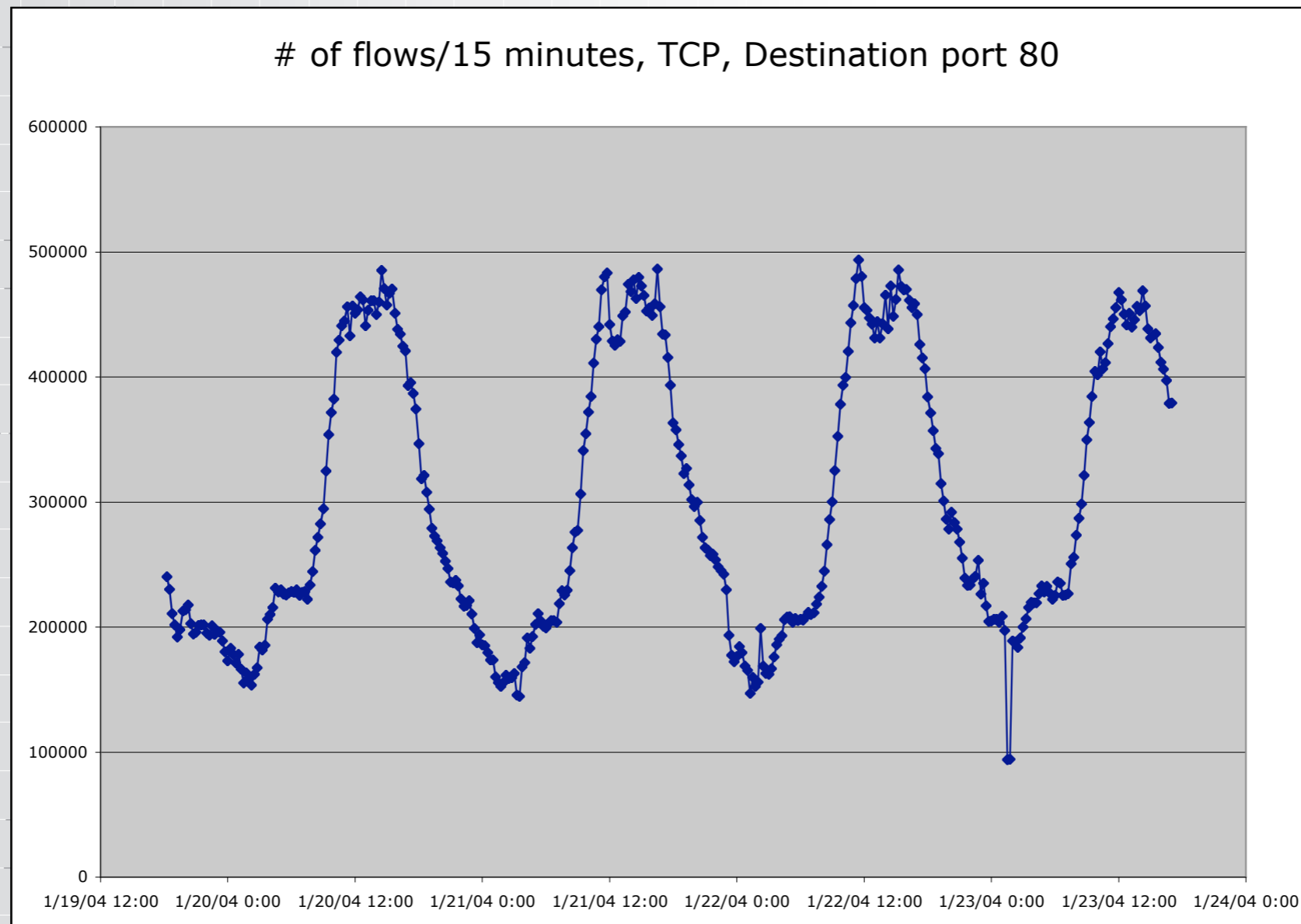
```
netflow=# select * from digested_tcp_dst where portnum=80 order by time_stamp;
```

time_stamp	portnum	count
2004-01-19 18:15:04-05	80	240206
2004-01-19 18:30:00-05	80	230117
2004-01-19 18:45:00-05	80	210618
2004-01-19 19:00:00-05	80	201564
2004-01-19 19:15:00-05	80	191944

etc...



# Added Value



Not hugely exciting, but it was free.



# Performance?

- We were feeling quite pleased with ourselves at this point.
- It quickly became apparent that we were going to have performance issues.
  - Storage volume
  - Raw processing power - operations were taking much longer than we might have liked.



# Performance?

- How could we improve the situation?
  - We're having to scan an immense amount of data
  - Indexes makes searching data fast!
  - The First Law of Database Thermodynamics
  - Indexes make inserting data slow!
  - We're having to insert an immense amount of data



# Performance issues?

- What else could we try?
  - Reduce the amount of raw data that we have to process
    - Keep a rotating archive
  - This creates more problems than it solved.
    - You have to delete data; this is expensive.
    - You have to clean up; this is also expensive.
  - Bigger, faster hardware, distribute tasks.
    - This would be nice.



# What else?

- We can also try and reduce the volume of data and/or improve the efficiency of the database with better normalization
  - Data common across records
  - Single flows generate records from multiple routers.
  - More expense



pervasivet<sup>technology</sup>labs  
AT INDIANA UNIVERSITY

*www.pervasivet<sup>technology</sup>labs.iu.edu*

# What else?

- We can do as much pre-processing ahead of time
  - You can never cover all the bases
  - You never know what you're going to be looking for
  - Waste time generating products that you'll never use.



pervasivetechlabs  
AT INDIANA UNIVERSITY

[www.pervasivetechlabs.iu.edu](http://www.pervasivetechlabs.iu.edu)

# What else?

- \$\$\$
  - Distributed setup for pre-processing, raw archive, post-processing.
  - The sky is the limit.



# Conclusion

- Netflow has the potential to be incredibly useful for network security and performance maintenance
  - Although it does have limitations
- Maintaining an archive that can be quickly and efficiently queried would be super
  - Especially if we can do compound queries
  - But we need **all** the data - you never know what you're going to be looking for.
  - But that was harder than we hoped.