# flowstatd - 那五年

*Kudo Chien*

# Kudo Chien

- CCUCSIE 2002-2008 BS+MS (GAIS Lab)
- CNA
- 曾工作於 Trend Micro、Waveface
- 現任 biideal CTO

# Kudo Chien

- 打雜
  - UN*X system programming
  - Windows programming
  - Network programming
  - Cloud / Web backend
  - Web frontend
  - Browser extension development
  - DevOps
  - Hacking
  - iOS
  - Android
- Debug

# 在 biideal 我們沒有辦不到的事

*誤*

# flowstatd 是?

## 中正大學宿網流量統計

By CNA 校園網路策進會

流量排行 ⋮ 宿網首頁 ⋮ 本系統資料約每 2 分鐘更新。

140.123.235.217

| Rank No. | IP | 總流量 (MB) |
|---|---|---|
| 1 | 140.123.218.213 | 29997.886393 |
| 2 | 140.123.238.14 | 12273.797497 |
| 3 | 140.123.236.80 | 10593.71641 |
| 4 | 140.123.215.101 | 8436.904972 |
| 5 | 140.123.218.180 | 7945.611323 |
| 6 | 140.123.224.138 | 7899.04346 |
| 7 | 140.123.213.185 | 7874.094663 |
| 8 | 140.123.236.43 | 7518.026353 |
| 9 | 140.123.215.28 | 7430.262412 |
| 10 | 140.123.224.148 | 7417.0754 |
| 11 | 140.123.236.44 | 7265.301674 |
| 12 | 140.123.219.118 | 7000.979501 |
| 13 | 140.123.224.166 | 6720.358048 |

July 2014

| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | | |

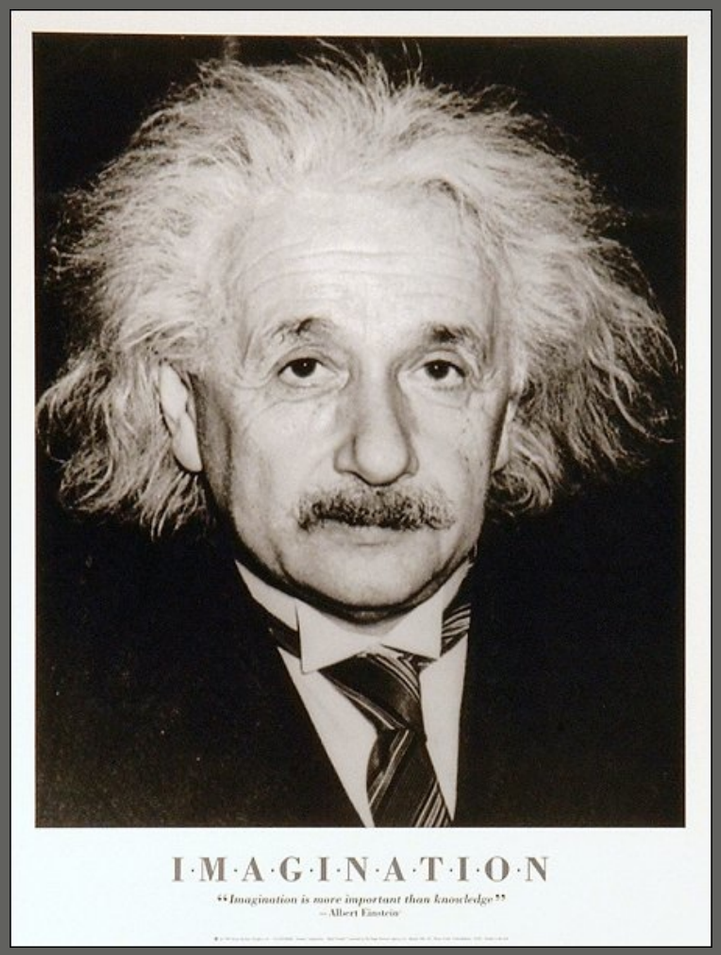
*Image source: https://www.flickr.com/photos/horiavarlan/4273913966*

# 這樣的系統需要什麼樣的機器來跑

*Image source: https://www.flickr.com/photos/horiavarlan/4273913966*

# memory / disk 使用量多大

The difference between genius and stupidity is that genius has its limits.



因為有限制

才得以出類拔萃

# Netflow introduction

- From Cisco
- Analyze traffic
  - SRC/DST IP
  - SRC/DST Port
  - TOC
  - IP Protocol

# 宿網流量統計 v1

- 來自交大 open source 的版本
  - flow-tools + Perl script
- 每小時”重頭”算一次統計
- 是網管的災難，使用者的福音

# 宿網流量統計 v2

- 由月光小俠 Eintisy 學長用 PHP 重寫的版本
- “累加”流量解決了第一版的問題
- 慢慢還是撐不住全校的流量
- 兩小時跑一次，網路速度越來越快，兩小時可以衝很多 GB

# 年少輕狂的 MySQL 時代

- 不管 3721，往 MySQL 丟就對了 *誤*
- MySQL 大神會幫你管理一切事務
- Malicious Detection

# 年少輕狂的 MySQL 時代

- 以 CCU 全校流量來說，倒進 MySQL 平均每小時佔用 Disk 1xx MB

# 吳昇老師的教誨

- Data Structure
- 對資料本質的掌握與計算
- Hash Hash Hash

# 重視統計流量的本質

- 累加流量
- IP address hash table - 一個蘿蔔一個坑


# 那五年 2007~2012

*2009 才開始在這個 project 用 git *悶**

作者 Kudo（東樓江水西樓月）　　站內 PJ_NetManage
標題 [idea] Flow Daemon
時間 2007/03/24 Sat 14:25:37

A daemon that save all flow statistic in memory

---
A. structure

```
struct hostflow {
    struct in_addr sin_addr;
    unsigned long long int hflow[24][2];        // 24 hours,  Upload/Download
    unsigned long long int nflow[3];            // flow now,Upload/Download/Sum
};

sizeof(struct hostflow) = 412
254 (Class C)   x   3 (一棟最多三個網段)   x    10 (宿舍數)     = 7620

412 x 7620 = 3,139,440  => 約 3.1 MB
```

統計全宿網一天的流量只需要

3.1 MB

全中正 Class B 的流量只需要

25.7 MB

# Flow daemon

# All in memory

# Real time

# Hash function v1

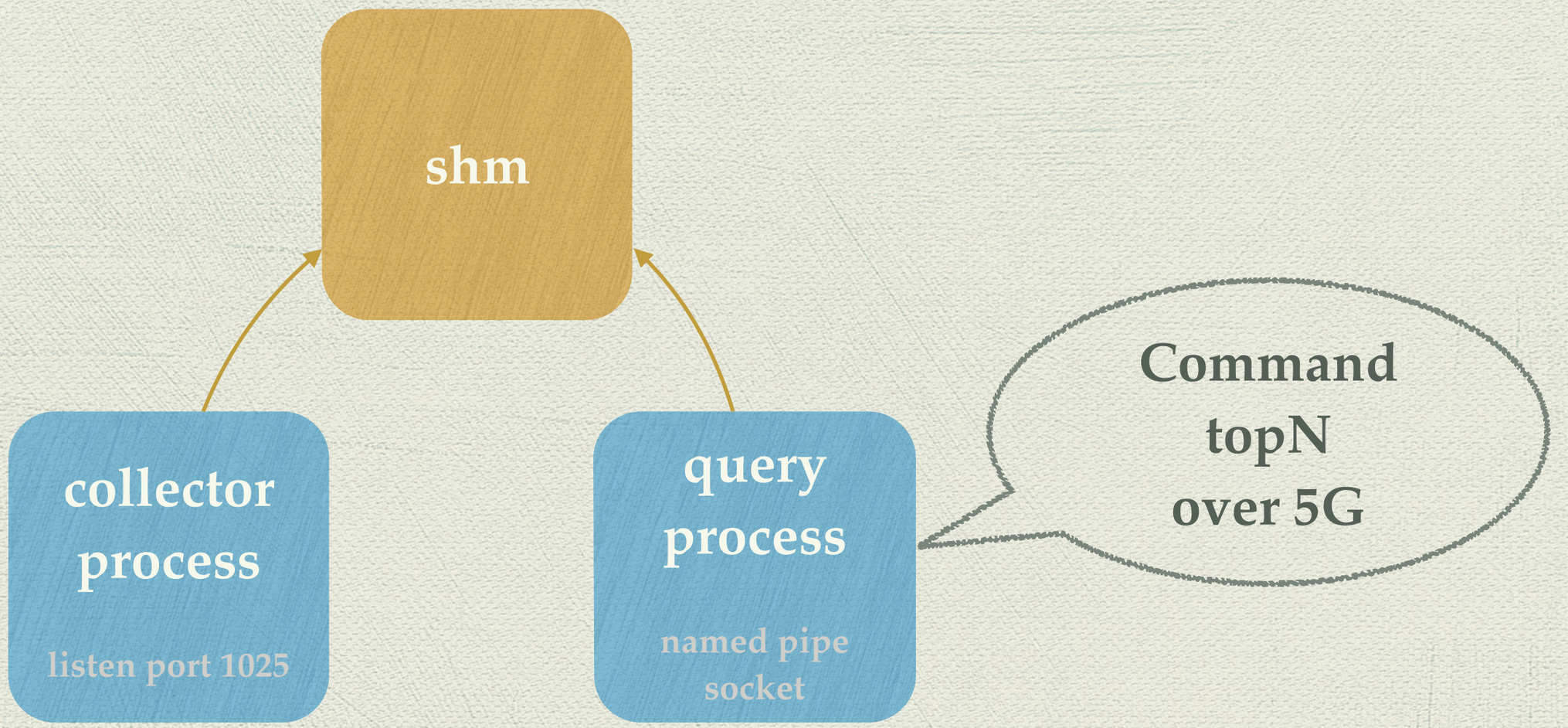
```
/* for dormnet ips */
int hash(struct in_addr sin_addr)
{
  char *ptr;
  char buf[256];
  int ip3, ip4;

  strncpy(buf, inet_ntoa(sin_addr), 255);

  ptr = strtok(buf, ".");          // ip1
  strtok(NULL, ".");               // ip2

  ptr = strtok(NULL, ".");
  ip3 = (int) strtol(ptr, (char **) NULL, 10);

  ptr = strtok(NULL, ".");
  ip4 = (int) strtol(ptr, (char **) NULL, 10);

  if (ip3 >= 211 && ip3 <= 225)
    return (((ip3 - 211) * 254) + ip4);
  else if (ip3 >= 232 && ip3 <= 240)
    return (((ip3 - 232 + 15) * 254) + ip4);
}
```

Image source: https://www.flickr.com/photos/sixybeast/8690039773/

# Hash function v2

```
B. hash function

之前的太麻煩了，換一個 @@

#define IP_PREFIX                   0x00007b8c                  // 140.123.0.0/16

int hash(struct NF_record *record)
{
  if ((record->srcaddr & 0x0000ffff) == IP_PREFIX)
  {
    return ntohs(record->srcaddr >> 16);
  }
  else if ((record->dstaddr & 0x0000ffff) == IP_PREFIX)
  {
    return ntohs(record->dstaddr >> 16);
  }

  return 0;
}
```

# Architecture v1



query process

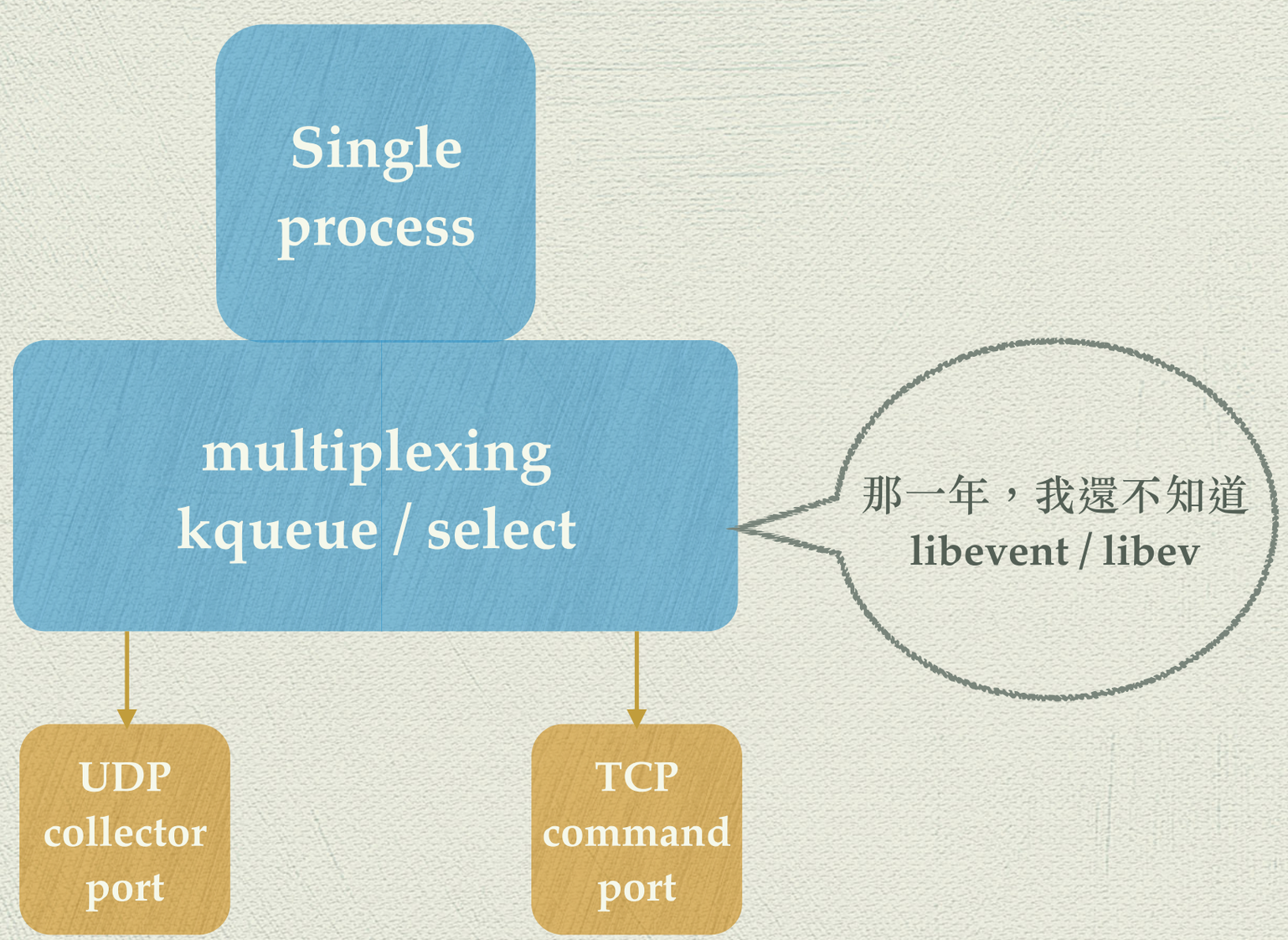
~~named pipe~~
socket

@WanCW
<(_ _)>

```
→ WanCW 推: IPC -> socket --> distributed system. (Y)                    07/03/26
→ Kudo 推: (YY)                                                          07/03/26
```

Image source: https://www.flickr.com/photos/sixybeast/8690039773/

# Architecture v2



# Object Oriented Programming

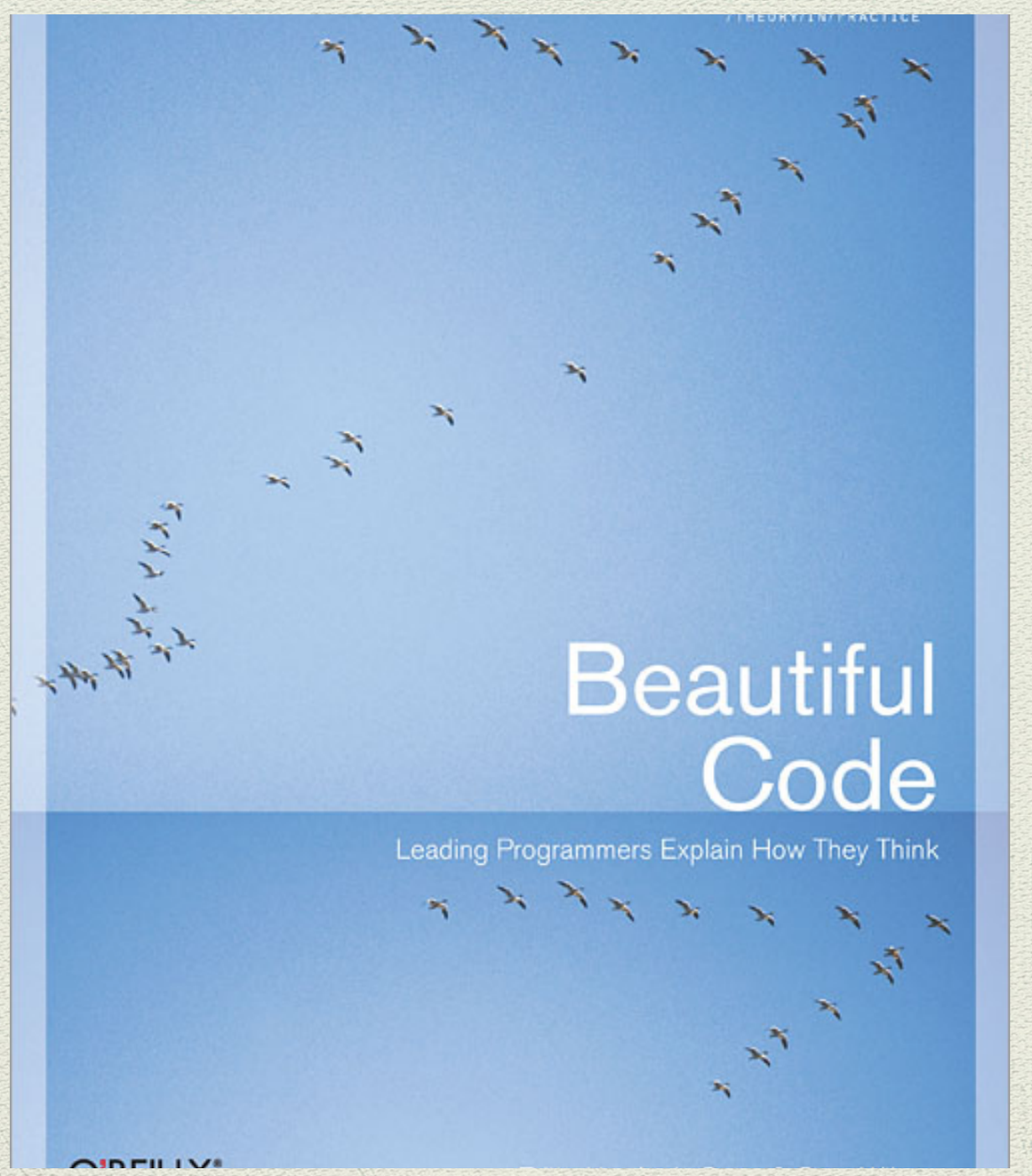
- Trained from Trend Micro
- 常見的好習慣是把 shared code 拆成 functions
  OOP 則更進一步把 shared behaviors 詮釋成共同的 interfaces
  - 以上是本人不負責任亂掰的說法 *誤*

"All problems in computer science can be solved by another level of indirection"

*–Butler Lampson*

# Object Oriented C

- Abstract + struct + function pointer
- select()/kqueue multiplexer
- Netflow v5/v9 handlers



```c
typedef struct _MultiplexerFunc_t MultiplexerFunc_t;
struct _MultiplexerFunc_t {
    int (*Init)(MultiplexerFunc_t *this);
    int (*UnInit)(MultiplexerFunc_t *this);

    int (*IsActive)(MultiplexerFunc_t *this, int fd);
    int (*AddToList)(MultiplexerFunc_t *this, int fd);
    int (*RemoveFromList)(MultiplexerFunc_t *this, int fd);
    int (*Wait)(MultiplexerFunc_t *this);
};
```

```c
#ifdef USE_KQUEUE
typedef struct _kqueueMultiplexer_t {
    MultiplexerFunc_t funcs;

    int kqFd;
    struct kevent evlist[MAX_MONITOR_FD_COUNT];
    struct kevent chlist[MAX_MONITOR_FD_COUNT];
    unsigned int monitorFdCount;
} kqueueMultiplexer_t;

int kqueueInitImpl(MultiplexerFunc_t *this);
int kqueueUnInitImpl(MultiplexerFunc_t *this);
int kqueueIsActiveImpl(MultiplexerFunc_t *this, int fd);
int kqueueAddToListImpl(MultiplexerFunc_t *this, int fd);
int kqueueRemoveFromListImpl(MultiplexerFunc_t *this, int fd);
int kqueueWaitImpl(MultiplexerFunc_t *this);
MultiplexerFunc_t *kqueueNewMultiplexer();
int kqueueFreeMultiplexer(MultiplexerFunc_t *this);

#endif
```

```c
typedef struct _selectMultiplexer_t selectMultiplexer_t;
struct _selectMultiplexer_t {
    MultiplexerFunc_t funcs;

    fd_set evlist;
    fd_set chlist;
    unsigned int maxFd;
};

int selectInitImpl(MultiplexerFunc_t *this);
int selectUnInitImpl(MultiplexerFunc_t *this);
int selectIsActiveImpl(MultiplexerFunc_t *this, int fd);
int selectAddToListImpl(MultiplexerFunc_t *this, int fd);
int selectRemoveFromListImpl(MultiplexerFunc_t *this, int fd);
int selectWaitImpl(MultiplexerFunc_t *this);
MultiplexerFunc_t *selectNewMultiplexer();
int selectFreeMultiplexer(MultiplexerFunc_t *this);
```

```
MultiplexerFunc_t *selectNewMultiplexer()
{
    selectMultiplexer_t *multiplexer = (selectMultiplexer_t *) malloc(sizeof(selectMultiplexer_t));
    multiplexer->funcs.Init = selectInitImpl;
    multiplexer->funcs.UnInit = selectUnInitImpl;
    multiplexer->funcs.IsActive = selectIsActiveImpl;
    multiplexer->funcs.AddToList = selectAddToListImpl;
    multiplexer->funcs.RemoveFromList = selectRemoveFromListImpl;
    multiplexer->funcs.Wait = selectWaitImpl;
    return &(multiplexer->funcs);
}
```

```
int selectInitImpl(MultiplexerFunc_t *this)
{
    selectMultiplexer_t *multiplexer = (selectMultiplexer_t *)this;
    multiplexer->maxFd = 0;
    FD_ZERO(&multiplexer->evlist);
    FD_ZERO(&multiplexer->chlist);

    return 1;
}
```

```
#ifdef USE_KQUEUE

#define NewMultiplexer(...) kqueueNewMultiplexer(__VA_ARGS__)
#define FreeMultiplexer(...) kqueueFreeMultiplexer(__VA_ARGS__)

#else

#define NewMultiplexer(...) selectNewMultiplexer(__VA_ARGS__)
#define FreeMultiplexer(...) selectFreeMultiplexer(__VA_ARGS__)

#endif
```

# Usage in caller is simple

```
multiplexer = NewMultiplexer();
if (multiplexer->Init(multiplexer) == 0)
    Diep("Multiplexer Init() failed");

multiplexer->AddToList(multiplexer, netflowSockFd);
multiplexer->AddToList(multiplexer, flowstatdSockFd);
```

# Over design 之 container_of

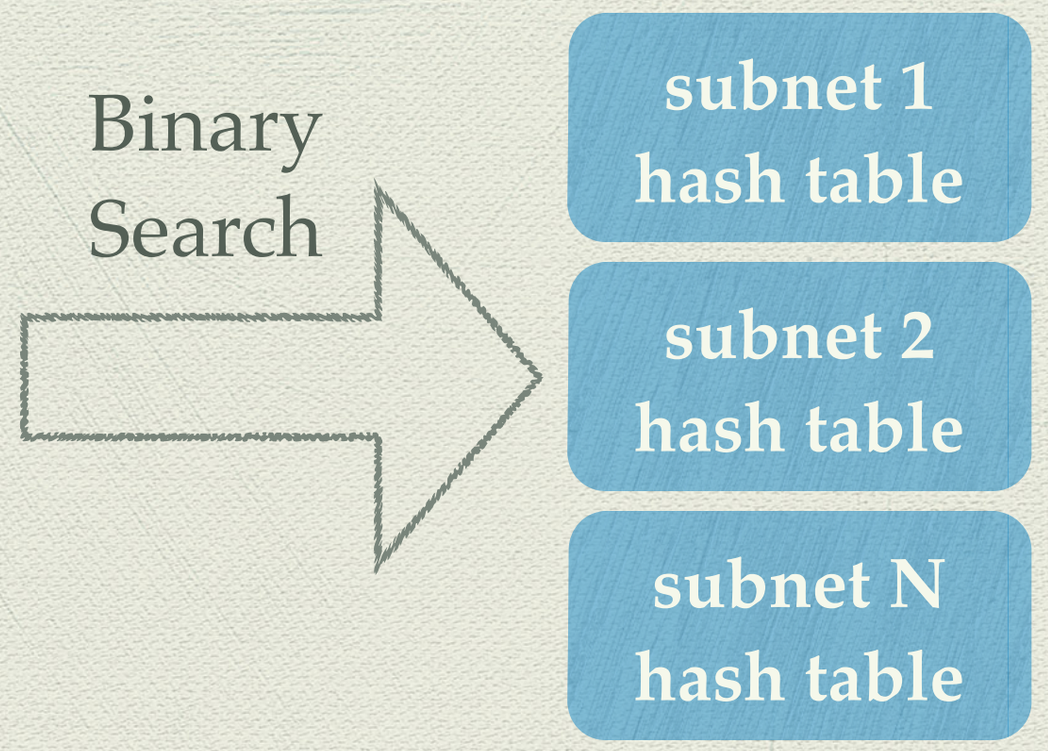
```c
/**
 * container_of - cast a member of a structure out to the containing structure
 * @ptr:        the pointer to the member.
 * @type:       the type of the container struct this is embedded in.
 * @member:     the name of the member within the struct.
 *
 */
#define container_of(ptr, type, member) ({                      \
        const typeof( ((type *)0)->member ) *__mptr = (ptr);    \
        (type *)( (char *)__mptr - offsetof(type,member) );})
```

直接 cast
就好啦 囧

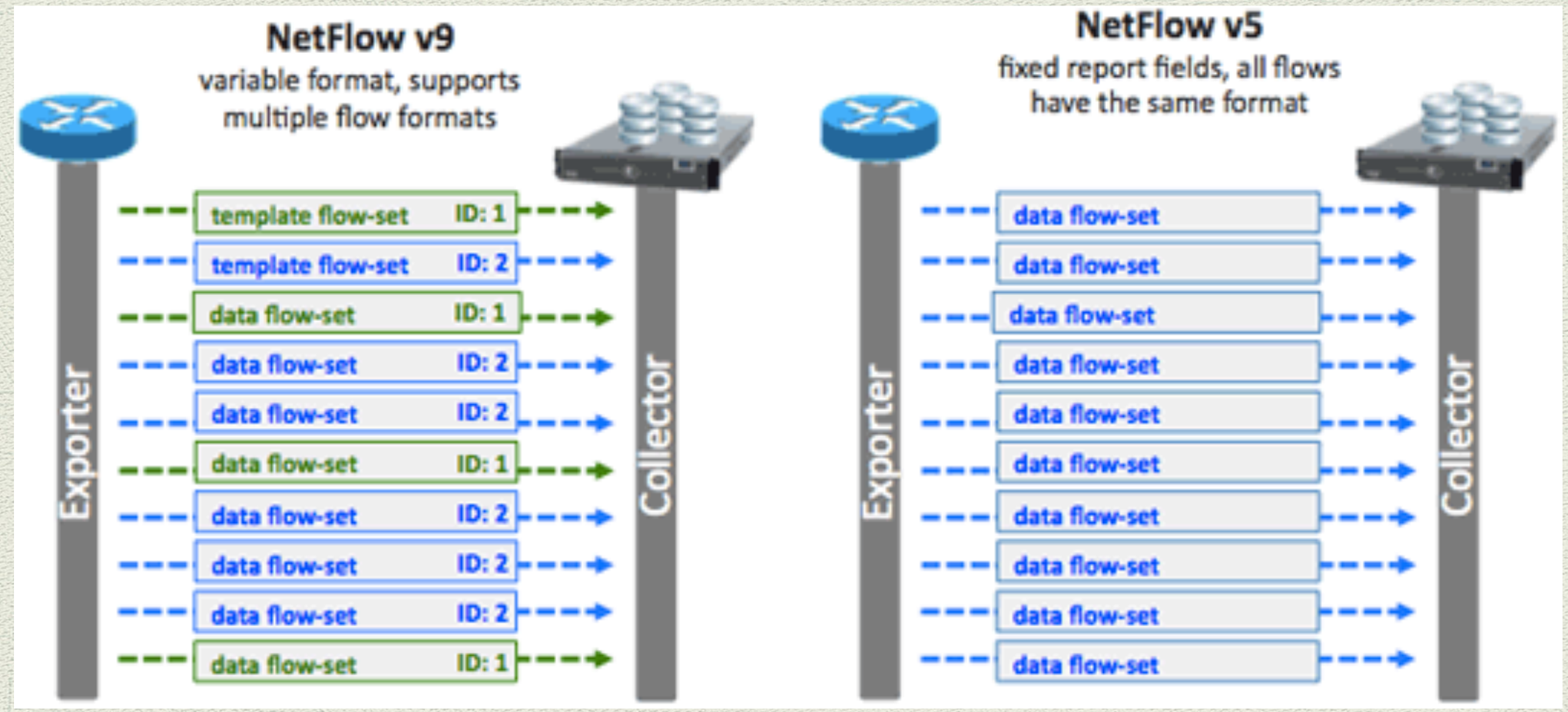
```c
int selectInitImpl(MultiplexerFunc_t *this)
{
    selectMultiplexer_t *multiplexer = container_of(this, selectMultiplexer_t, funcs);
    multiplexer->maxFd = 0;
    FD_ZERO(&multiplexer->evlist);
    FD_ZERO(&multiplexer->chlist);

    return 1;
}
```

# Multiple subnets

- 全校用一個 hash table 相對簡單
- 宿網 30 個 subnets 反而麻煩



# Netflow version 9




- template
- multiple source + multiple source id

Image source: http://www.lancope.com/blog/netflow-v5-vs-netflow-v9/

# Netflow version 9

- 多個 source ip 下面又可以有多個 source id
- 又想用 hash 又不想用太多 memory
- two hash tables
  - source table
  - template table

# Hash from two factors

```c
/* Data Type Definitions */
struct NF_V9_template_table_entry {
    uint16_t template_type;
    uint16_t field_count;
    uint16_t record_length;
    struct NF_V9_flowset_record fields[NF_V9_MAX_FIELDS_IN_TEMPLATE];
};

struct NF_V9_source_table_entry {
    in_addr_t sourceIp;
    uint32_t sourceId;
    struct NF_V9_template_table_entry *templatePtr;
};
```

```c
static int _HashingSourceInfo(in_addr_t sourceIp, uint32_t sourceId)
{
    // [0] Cantor pairing function
    unsigned long long key = (sourceIp + sourceId) * (sourceIp + sourceId + 1) / 2 + sourceId;

    // Ref: http://elliottback.com/wp/hashmap-implementation-in-c/
    // [1] Robert Jenkins 32 bit Mix Function
    key += (key << 12);
    key ^= (key >> 22);
    key += (key << 4);
    key ^= (key >> 9);
    key += (key << 10);
    key ^= (key >> 2);
    key += (key << 7);
    key ^= (key >> 12);

    // [2] Knuth's Multiplicative Hash
    key = (key >> 3) * 2654435761UL;

    return key % NF_V9_MAX_SOURCE_ENTRIES;
}
```

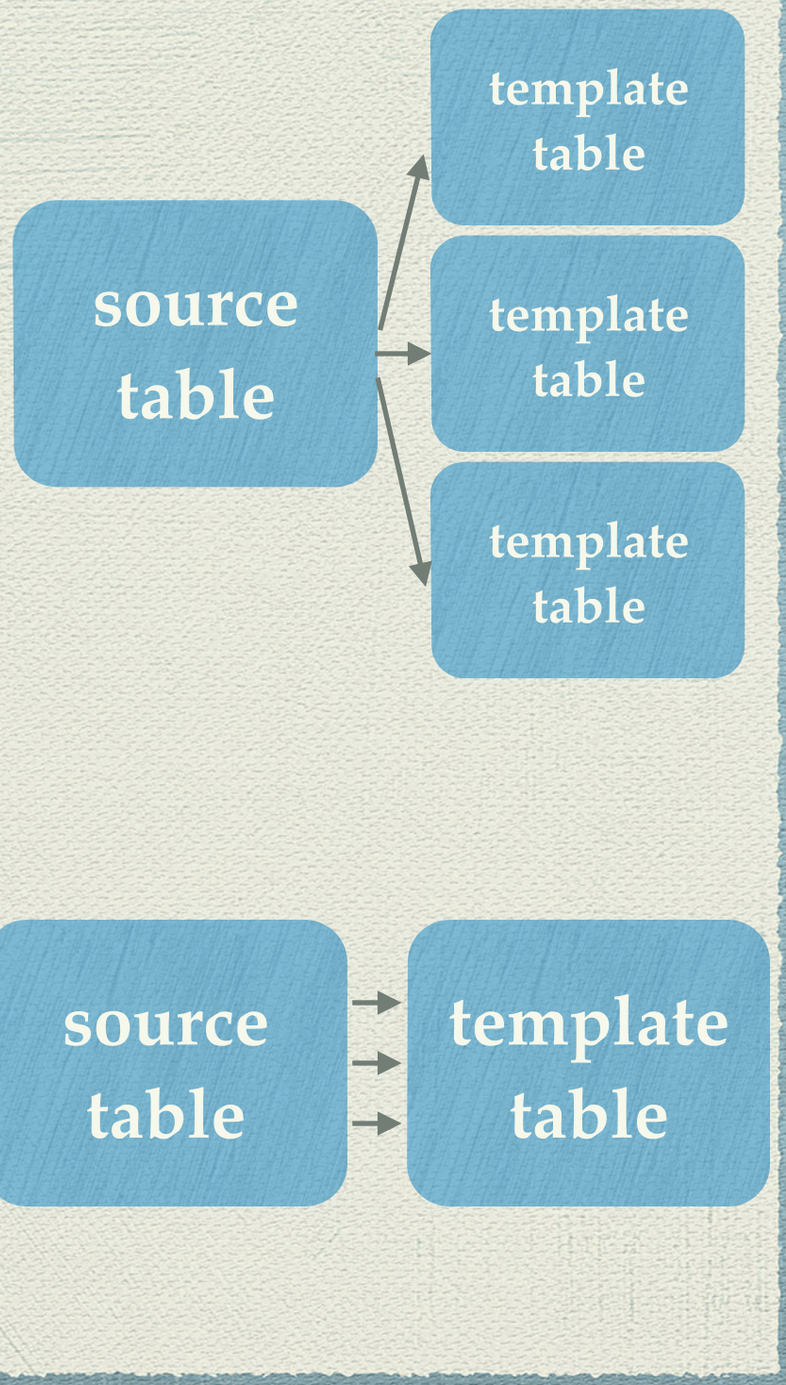


Image source: https://www.flickr.com/photos/sixybeast/8690039773/

# Netflow version 9 debugging

- 時好時壞的 bug 真難抓
- tcpdump -> pcap
- 對照組
  - pcap -> wireshark
  - pcap -> tcpreplay

# Misc

- JSON input command
- JSON output
- cmake
- logger
- gzopen() / gz*()

# flowstatd-frontend

- 圖像化才有感，偏偏我 UI 實作能力很差
  一年半載又過去了
- Open flash chart -> Google Chart API
- PHP -> Rails (純練習)

# Release

- 每每回頭來看，一點都不滿意，但是，好像是該讓他出門的時候了
- global variable
- clean code
- Many TODO

# fork me please

- https://github.com/Kudo/flowstatd
- https://github.com/Kudo/flowstatd-frontend

# 因為有限制

# 才得以出類拔萃

不只是設計系統，run startup 也是


Image source: https://www.flickr.com/photos/vernhart/1574355240/