Complexity Attack Resistant Flow Lookup Schemes for IPv6: A Measurement Based Comparison

David Malone and Josh Tobin.

2008-12-11

Hash Table

Lookup scheme to avoid cost of searching full list.

`carrot zuchinni` ... ...`apricot apple`

becomes:

- **a** `apricot, apple`
- **b** `banana`
- ...
- **z** `zuchinni`

Hash function $h(x)$, XOR typical. Cost: $O(N) \rightarrow O(N/H)$.

Algorithmic Attacks

Worst case rather than typical behaviour. (Crosby and Wallach, 2003).

Suppose attacker controls keys.

a `abduce, abducens, abducent, abduct, abduction, abductor` ...

b

.

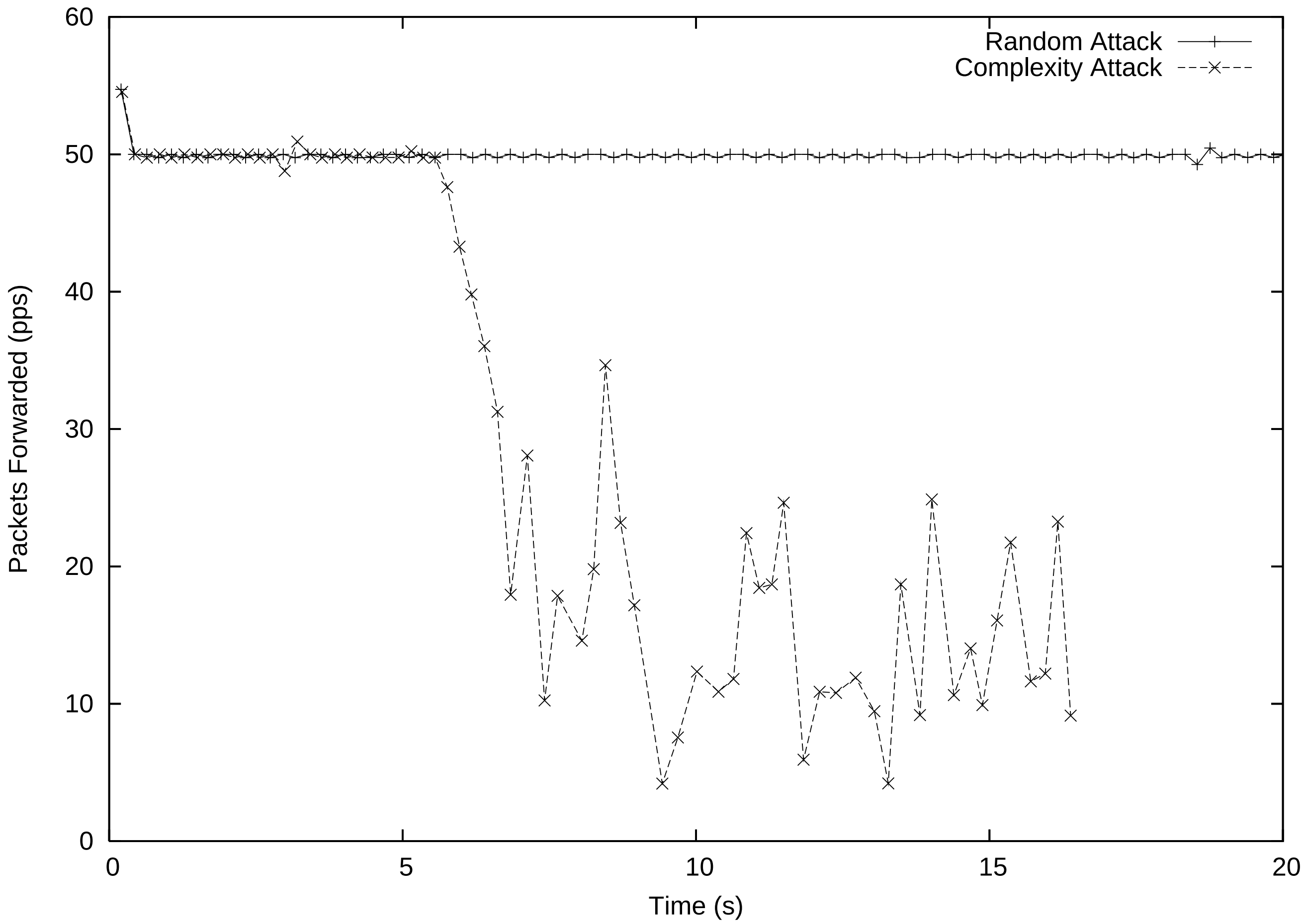
z

Attacker finds x_i so that $f(x_1) = f(x_2) = \ldots = f(x_i)$.

Flow Lookup

- Security applications often track IP flows.
- Lookup single flow by tuple (src IP, dst IP, src port, dst port).
- Hash table is one possible optimisation.
- For IPv4 96 bits of input.
- For IPv6 288 bits of input.
- Aim — `ipfw` for FreeBSD.
- Note inexact flow matching different!

Demonstration attack



Xor $h \leftarrow 0$

foreach $(byte[i])$ $h \leftarrow h \oplus byte[i]$

return h

DJB2 $h \leftarrow 5381$

foreach $(byte[i])$ $h \leftarrow 33 * h + byte[i]$

return h

XorSum $h \leftarrow 0$

foreach $(byte[i])$ $h \leftarrow h + (byte[i] \oplus K[i])$

return h

SumXor $h \leftarrow 0$

foreach $(byte[i])$ $h \leftarrow h \oplus (byte[i] + K[i])$

return h;

Universal $h \leftarrow 0$

foreach $(byte[i])$ $h \leftarrow h + K[i] * byte[i]$

return $h \mod 65537$

Pearson $h_1 \leftarrow h_2 \leftarrow 0$

foreach $(byte[i])$ $h_1 \leftarrow T_1[byte[i] \oplus h_1]$

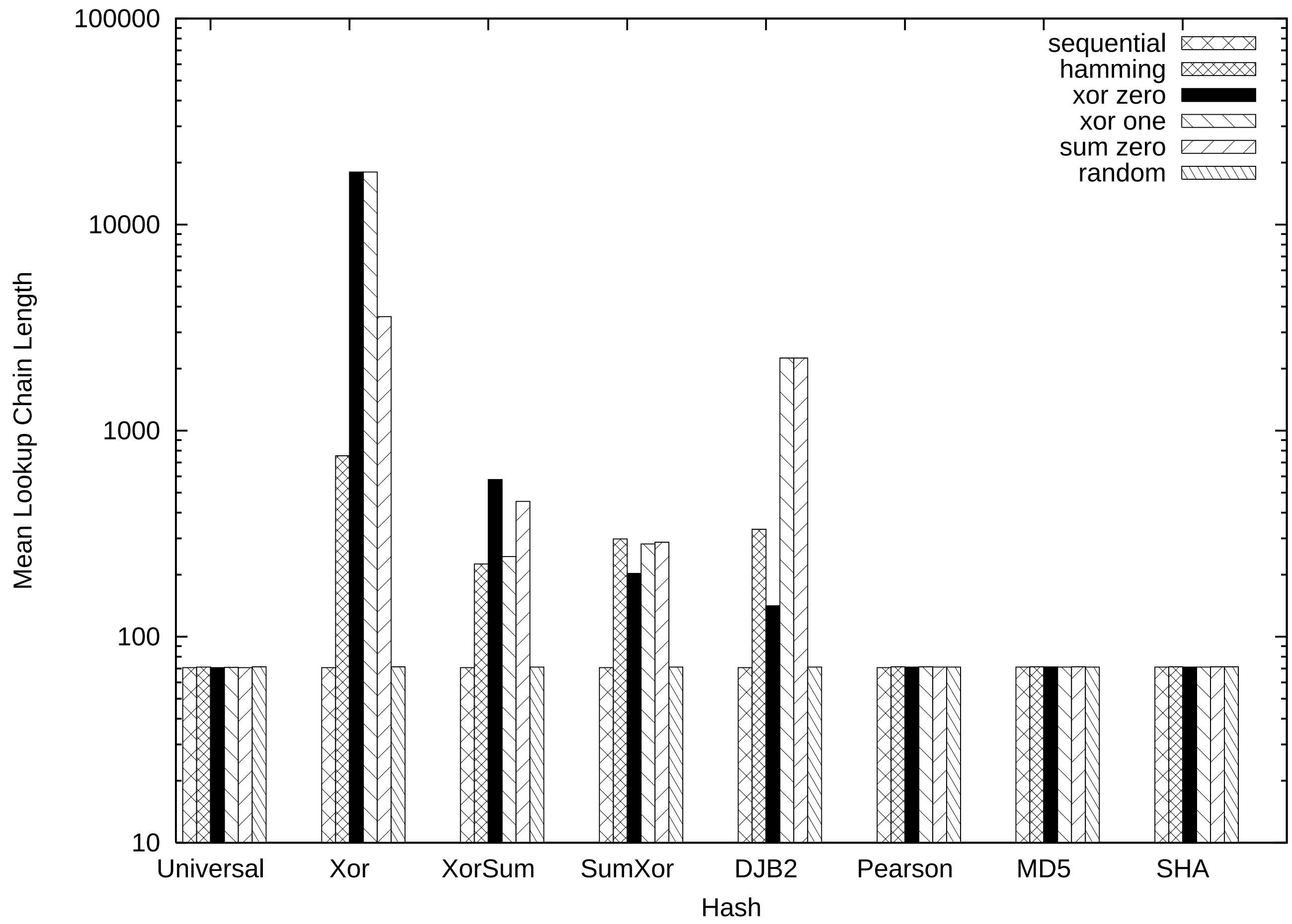
$h_2 \leftarrow T_2[byte[i] \oplus h_2]$

return $h_1 + h_2 * 256$

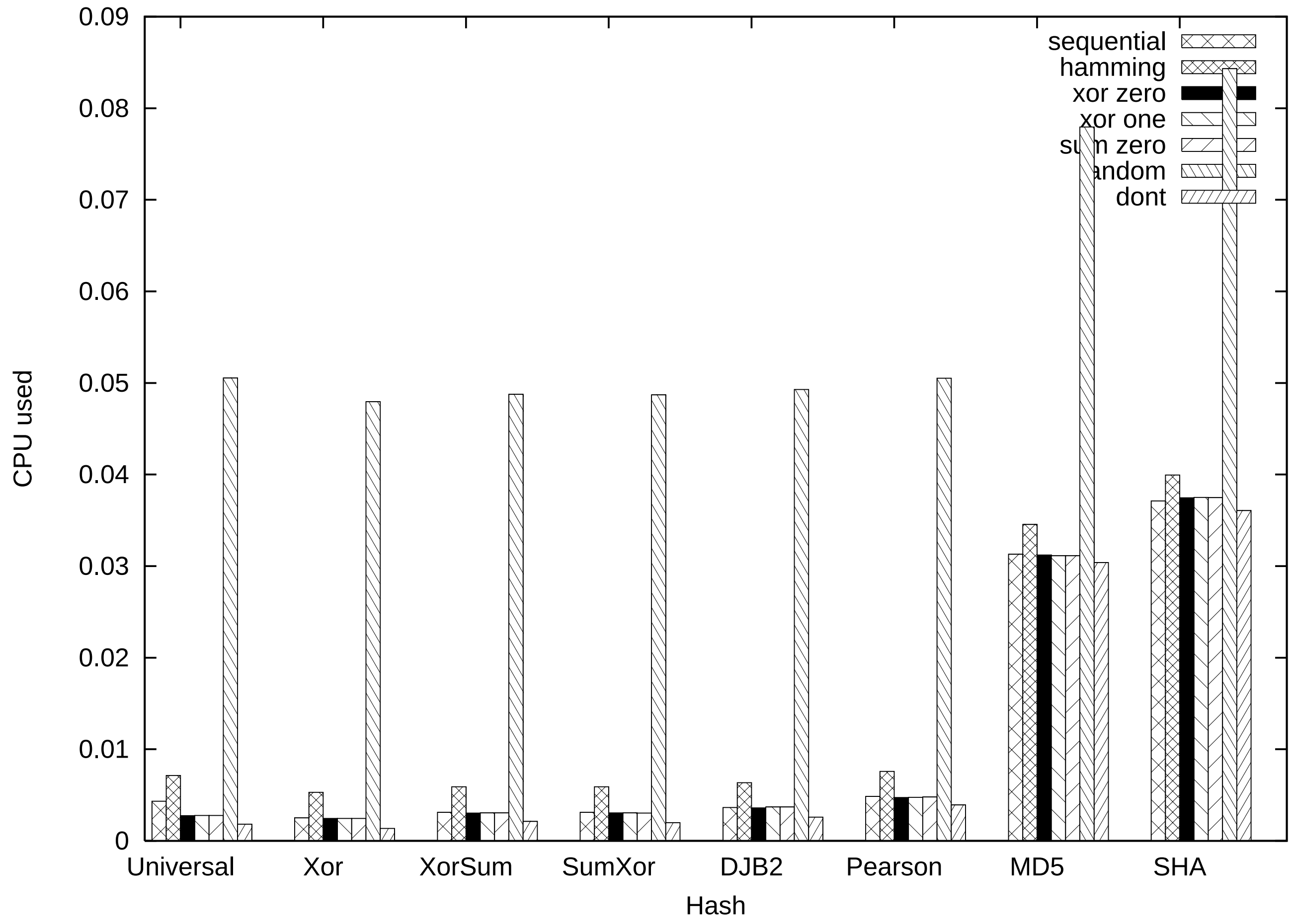
MD5 return two bytes of MD5(bytes)

SHA return two bytes of SHA(bytes)

Hash Chain Length



CPU Cost



0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
CPU used
sequential
hamming
xor zero
xor one
sum zero
random
dont
Universal
Xor
XorSum
SumXor
DJB2
Pearson
MD5
SHA
Hash

Other options

Don't need to use hash.

Tree Use lexical order to insert into tree.

Red/Black Tree Tree balanced by colouring.

Splay Tree Moves frequently accessed to top.

Treap Tree balanced using random heap.

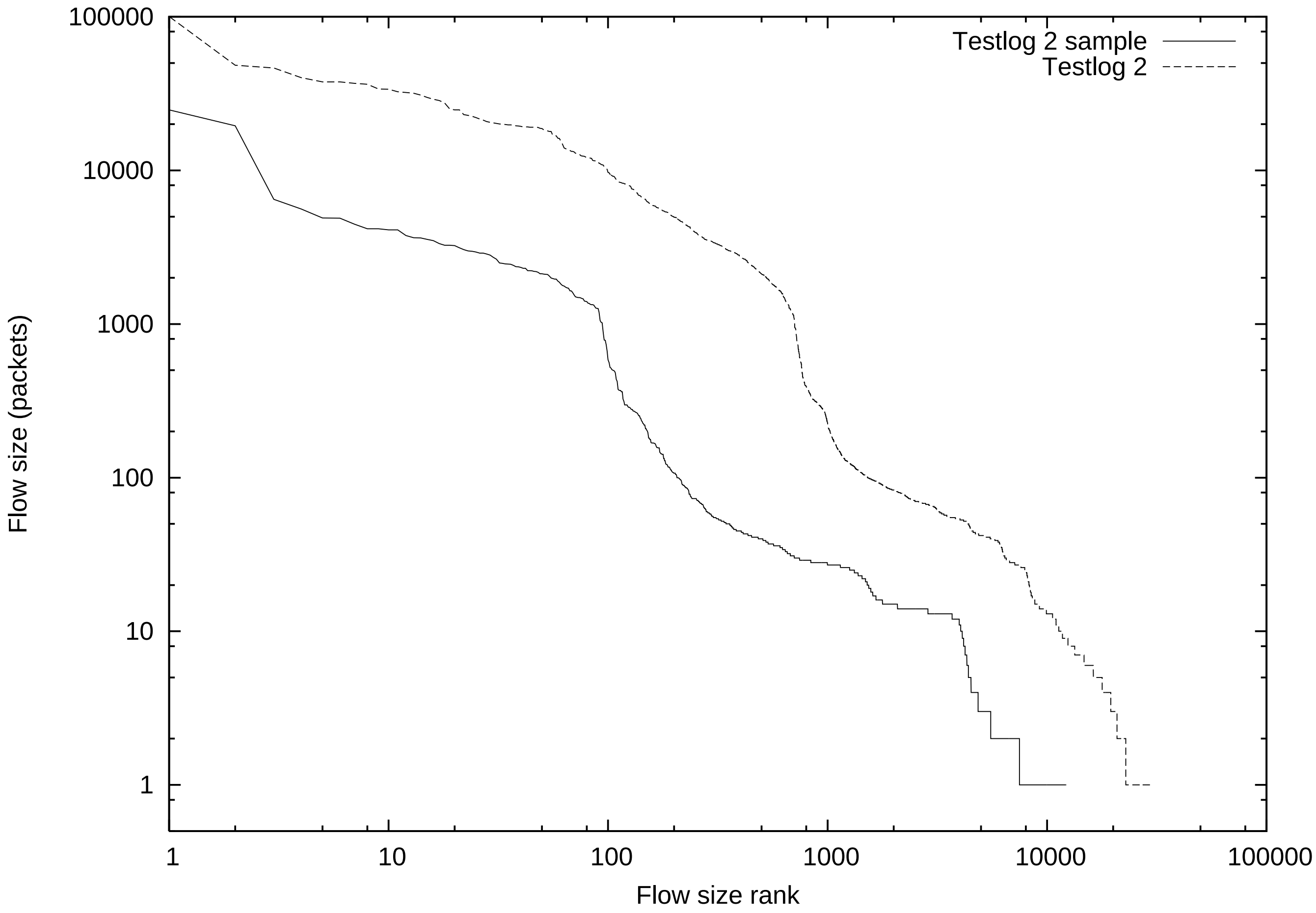
Tree is baseline (and subject to attack). Others are not (obviously) subject to attack.

Design Aims/Method

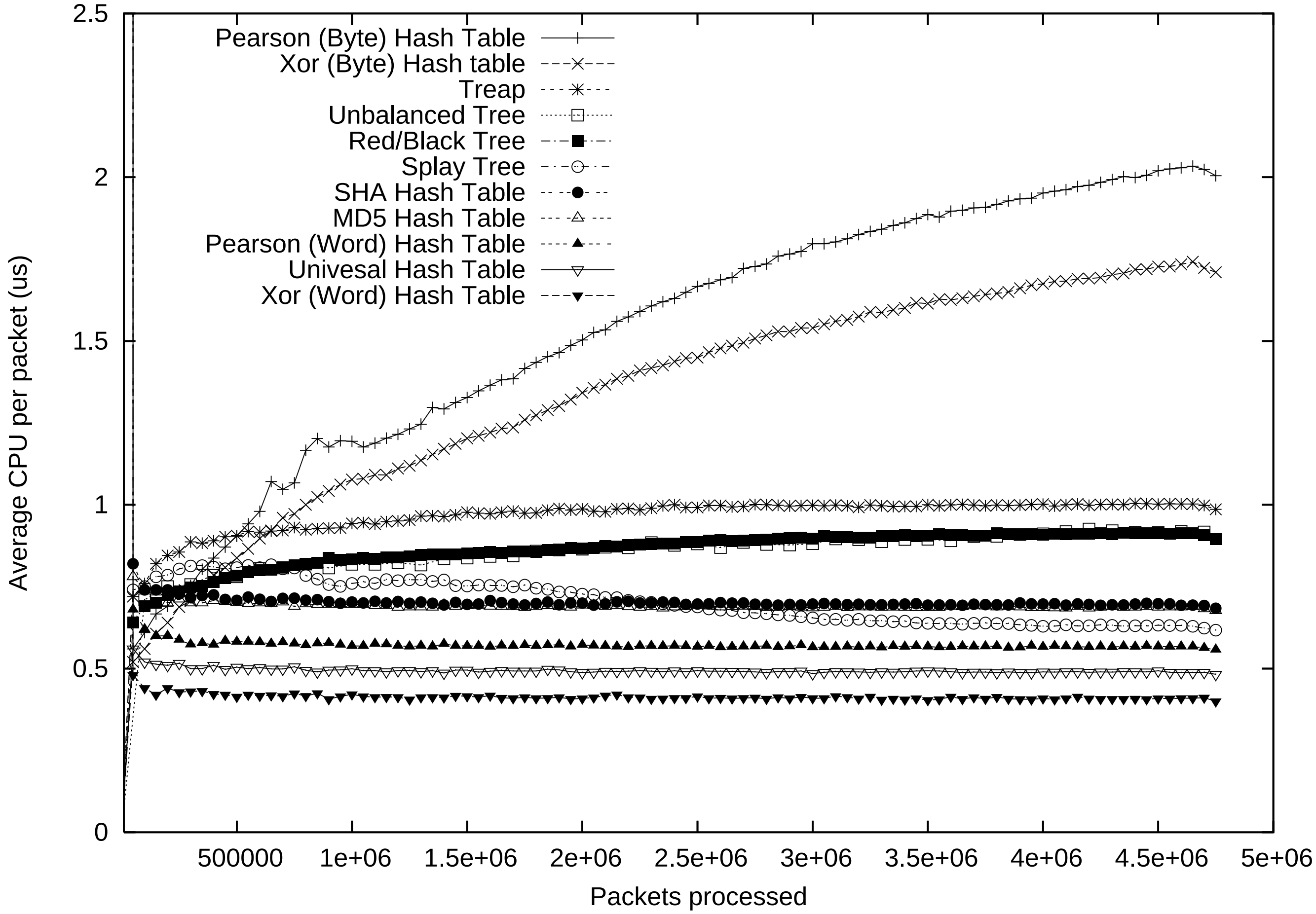
Want flow lookup to:

- Should perform OK under typical traffic.
- Should not degrade badly under attack.
- . . . typical performance depends on keys.
- . . . collect trace of traffic,
- . . . assess using pcap framework,
- . . . check performance in kernel.

Traffic Trace

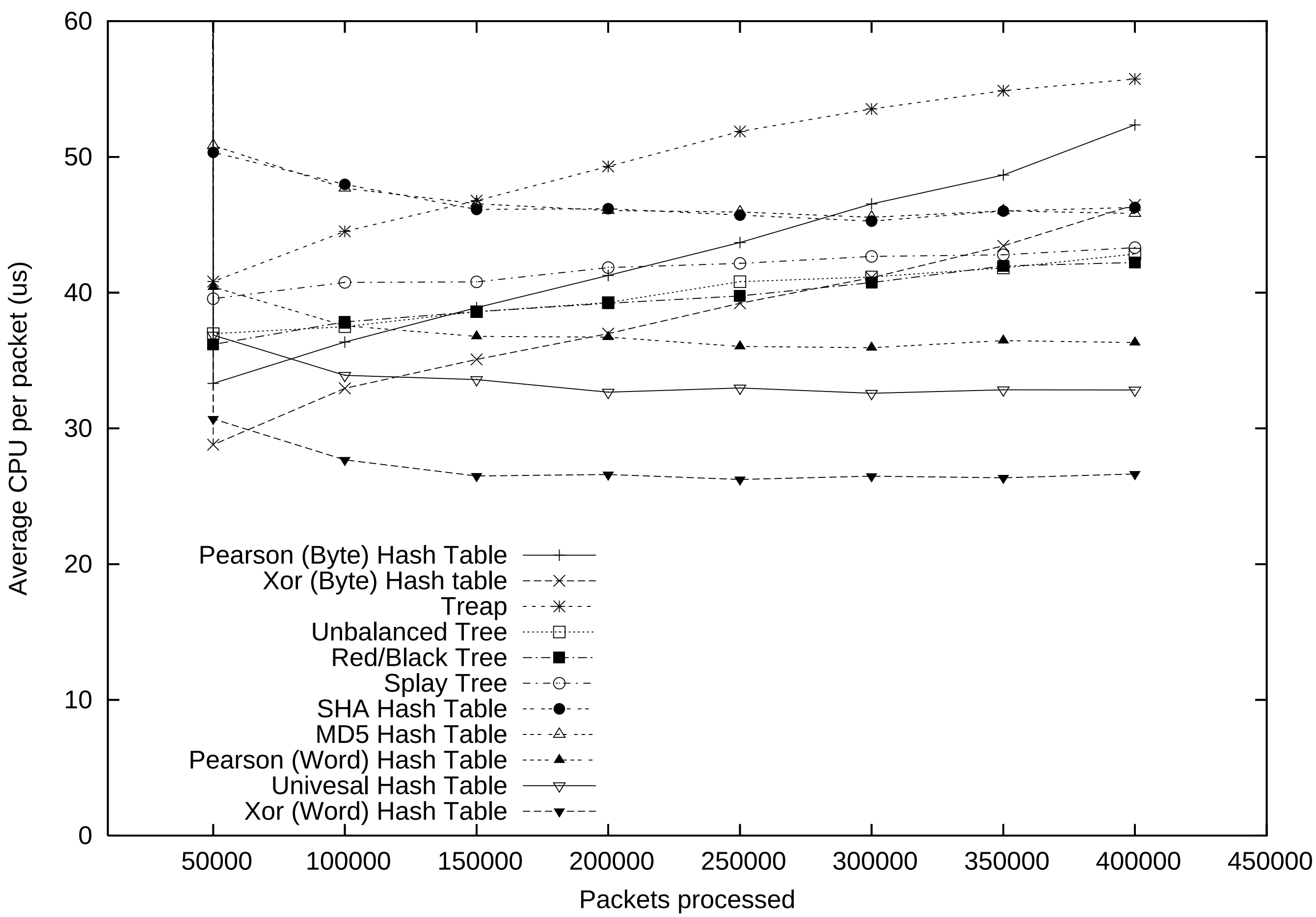


Big CPU

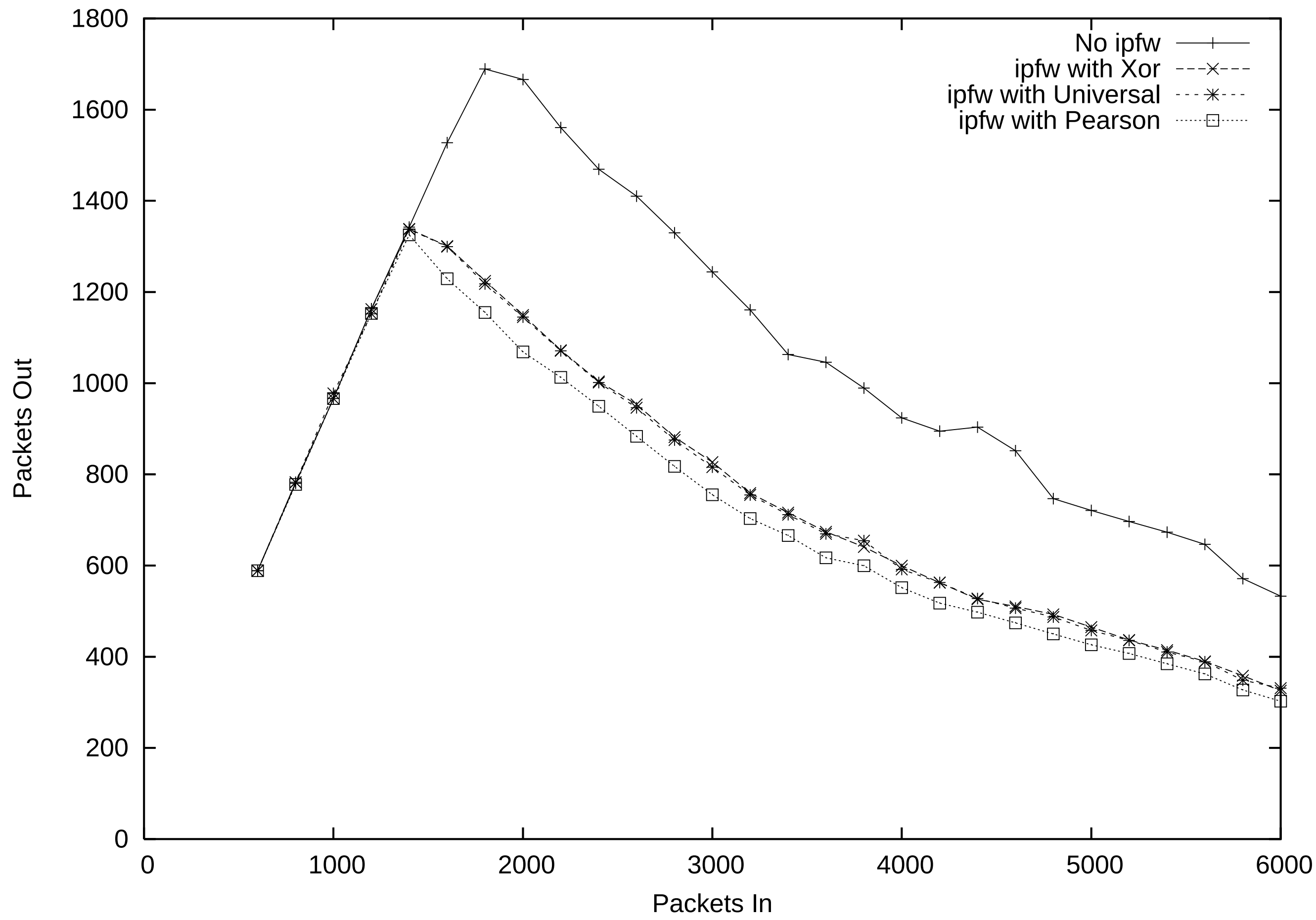


Pearson (Byte) Hash Table
Xor (Byte) Hash table
Treap
Unbalanced Tree
Red/Black Tree
Splay Tree
SHA Hash Table
MD5 Hash Table
Pearson (Word) Hash Table
Univesal Hash Table
Xor (Word) Hash Table
Average CPU per packet (us)
Packets processed
0
0.5
1
1.5
2
2.5
500000
1e+06
1.5e+06
2e+06
2.5e+06
3e+06
3.5e+06
4e+06
4.5e+06
5e+06

Small CPU


Average CPU per packet (us)
Packets processed
0
10
20
30
40
50
60
50000
100000
150000
200000
250000
300000
350000
400000
450000
Pearson (Byte) Hash Table
Xor (Byte) Hash table
Treap
Unbalanced Tree
Red/Black Tree
Splay Tree
SHA Hash Table
MD5 Hash Table
Pearson (Word) Hash Table
Univesal Hash Table
Xor (Word) Hash Table

Peak Forwarding



Summary

- Looked at flow lookup schemes.
- Wanted attach resistant scheme,
- ... with good typical performance.
- Future: get code into FreeBSD.
- Future: look at attacks on hashes.
- Future: new hashing schemes.