MOVING AWAY FROM OPENBGPD TO BIRD?

Apricot 2013, Singapore

Jimmy Halim

jhalim@ap.equinix.com

OPENBGPD IN A FLASH

- **3 processes**
  - Session Engine (SE): manages BGP sessions
  - Route Decision Engine (RDE): holds the BGP tables, takes routing decisions
  - Parent: enters routes into the kernel, starts SE and RDE
- **IPv4 and IPv6 in a single configuration**
- **BGP commands**
  - Using ‘bgpctl’ command for both IPv4 and IPv6

WORKING WITH OPENBGPD

The positive notes…

- **Stable with no related bug since upgrade to 4.8**
  - 4.3 has been bugged with bugs like BGP malformed attributes and IPv6 MD5 password errors
- **Provide the needed BGP functionality**
  - Transparent AS support
  - BGP community support for route manipulation
  - Support prefix filtering
- **Flexible BGP commands execution and configuration change**
  - Allow short form and help function from UNIX prompt

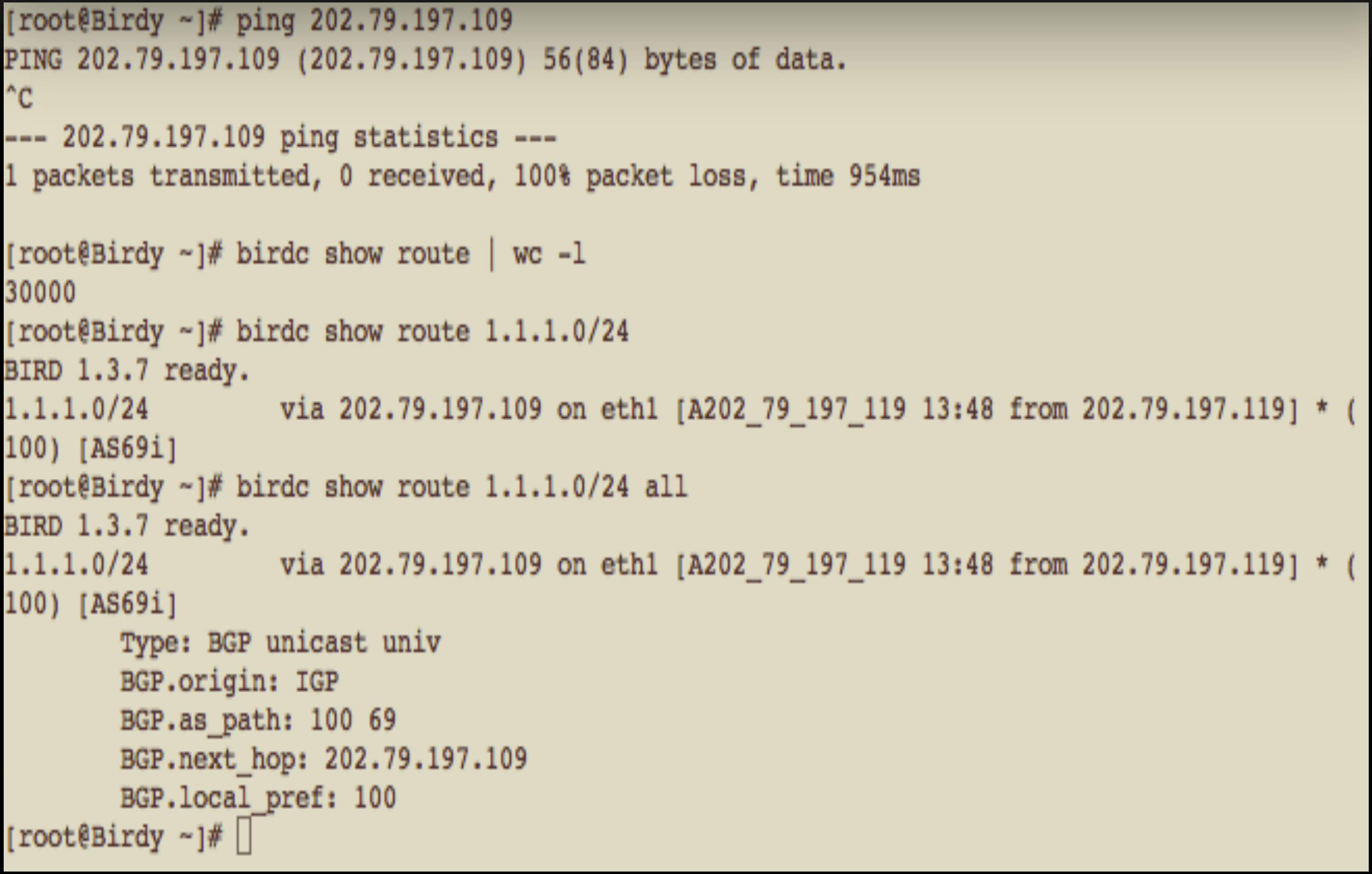
WORKING WITH OPENBGPD

The negative one…

- **No good in handling prefix filter**
  - Especially if we implement prefix filter per neighbor
    - Means more prefix filters to be created and checked
    - Example if we have 100 peers in IX, then there are at least 100 prefix filters need to be created and checked considering if each peer only have 1 prefix
  - Resulting in a very long routing convergence
    - More peers in IX
    - More routes
- **Problem with long routing convergence**
  - The routing convergence can take 2 hours, 6 hours, 12 hours, and even 1 day
  - The best route selection will not be optimal
  - Resulting in route blackhole!!

WORKING WITH OPENBGPD

Routing blackhole!!

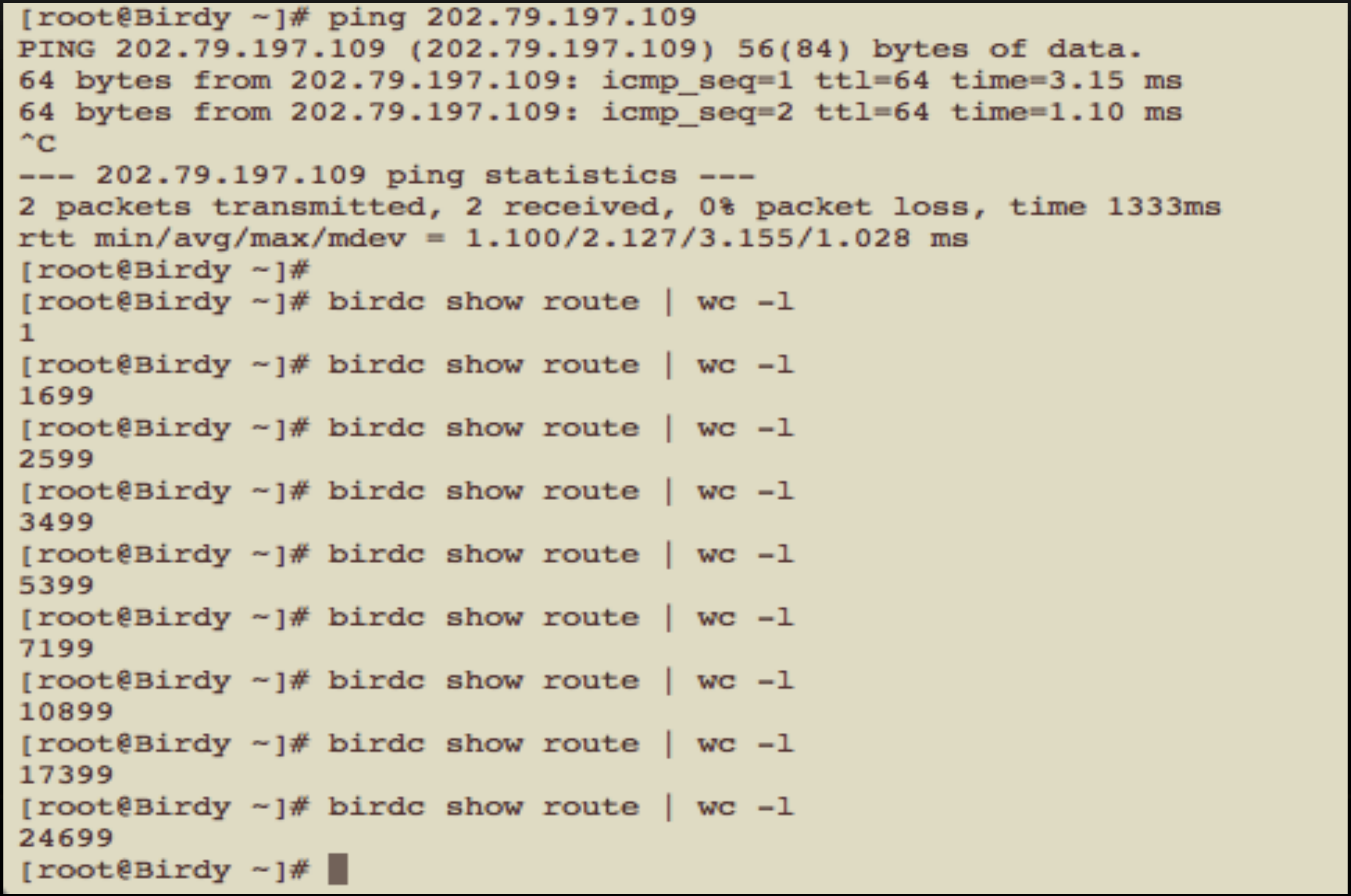


```
[root@Birdy ~]# ping 202.79.197.109
PING 202.79.197.109 (202.79.197.109) 56(84) bytes of data.
^C
--- 202.79.197.109 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 954ms

[root@Birdy ~]# birdc show route | wc -l
30000
[root@Birdy ~]# birdc show route 1.1.1.0/24
BIRD 1.3.7 ready.
1.1.1.0/24         via 202.79.197.109 on eth1 [A202_79_197_119 13:48 from 202.79.197.119] * (
100) [AS69i]
[root@Birdy ~]# birdc show route 1.1.1.0/24 all
BIRD 1.3.7 ready.
1.1.1.0/24         via 202.79.197.109 on eth1 [A202_79_197_119 13:48 from 202.79.197.119] * (
100) [AS69i]
        Type: BGP unicast univ
        BGP.origin: IGP
        BGP.as_path: 100 69
        BGP.next_hop: 202.79.197.109
        BGP.local_pref: 100
[root@Birdy ~]# 
```

WORKING WITH OPENBGPD

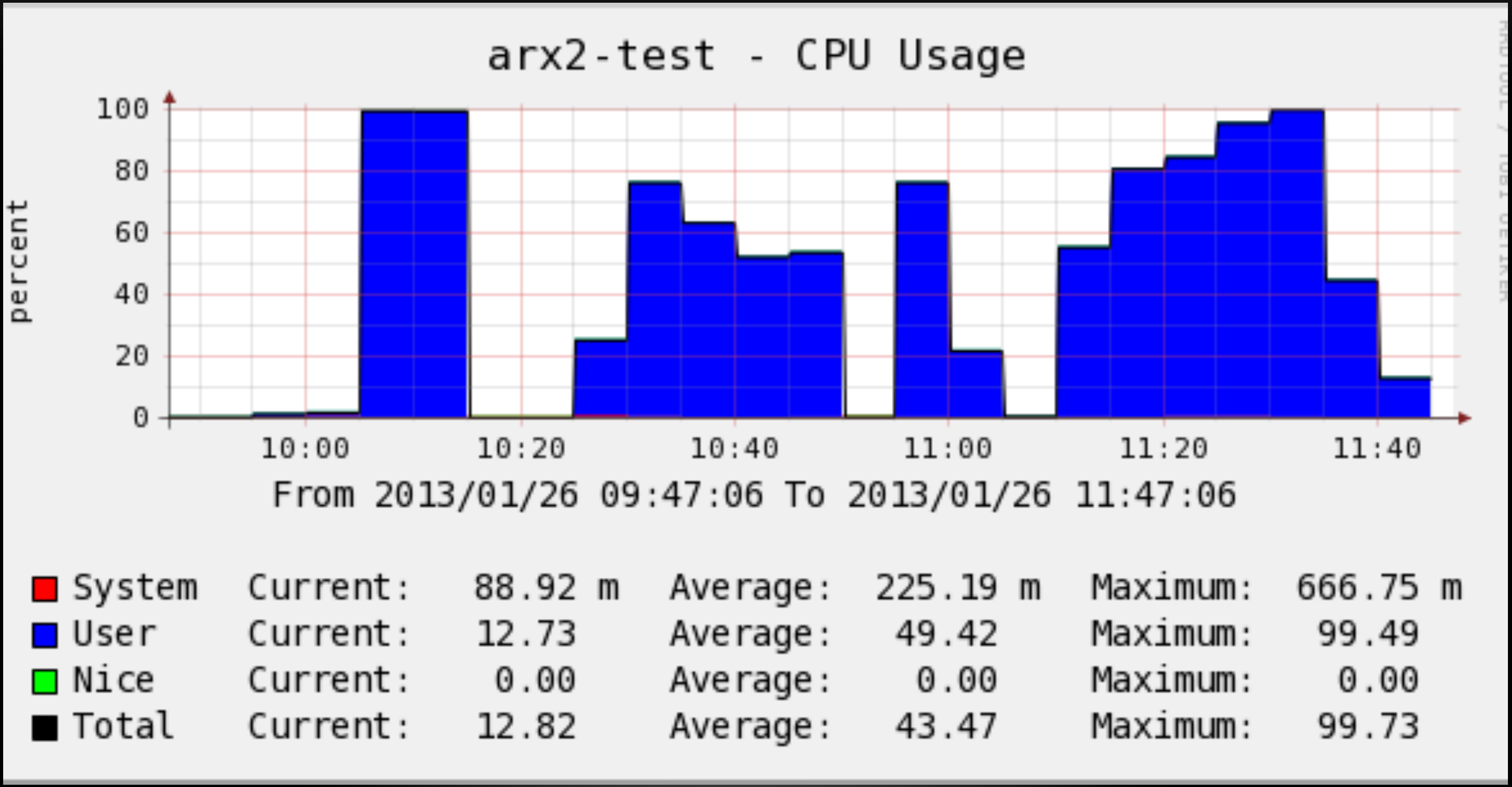
Long routing convergence…



```
[root@Birdy ~]# ping 202.79.197.109
PING 202.79.197.109 (202.79.197.109) 56(84) bytes of data.
64 bytes from 202.79.197.109: icmp_seq=1 ttl=64 time=3.15 ms
64 bytes from 202.79.197.109: icmp_seq=2 ttl=64 time=1.10 ms
^C
--- 202.79.197.109 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1333ms
rtt min/avg/max/mdev = 1.100/2.127/3.155/1.028 ms
[root@Birdy ~]#
[root@Birdy ~]# birdc show route | wc -l
1
[root@Birdy ~]# birdc show route | wc -l
1699
[root@Birdy ~]# birdc show route | wc -l
2599
[root@Birdy ~]# birdc show route | wc -l
3499
[root@Birdy ~]# birdc show route | wc -l
5399
[root@Birdy ~]# birdc show route | wc -l
7199
[root@Birdy ~]# birdc show route | wc -l
10899
[root@Birdy ~]# birdc show route | wc -l
17399
[root@Birdy ~]# birdc show route | wc -l
24699
[root@Birdy ~]# █
```

WORKING WITH OPENBGPD

High CPU…



🟥 System	Current: 88.92 m	Average: 225.19 m	Maximum: 666.75 m
🟦 User	Current: 12.73	Average: 49.42	Maximum: 99.49
🟩 Nice	Current: 0.00	Average: 0.00	Maximum: 0.00
⬛ Total	Current: 12.82	Average: 43.47	Maximum: 99.73

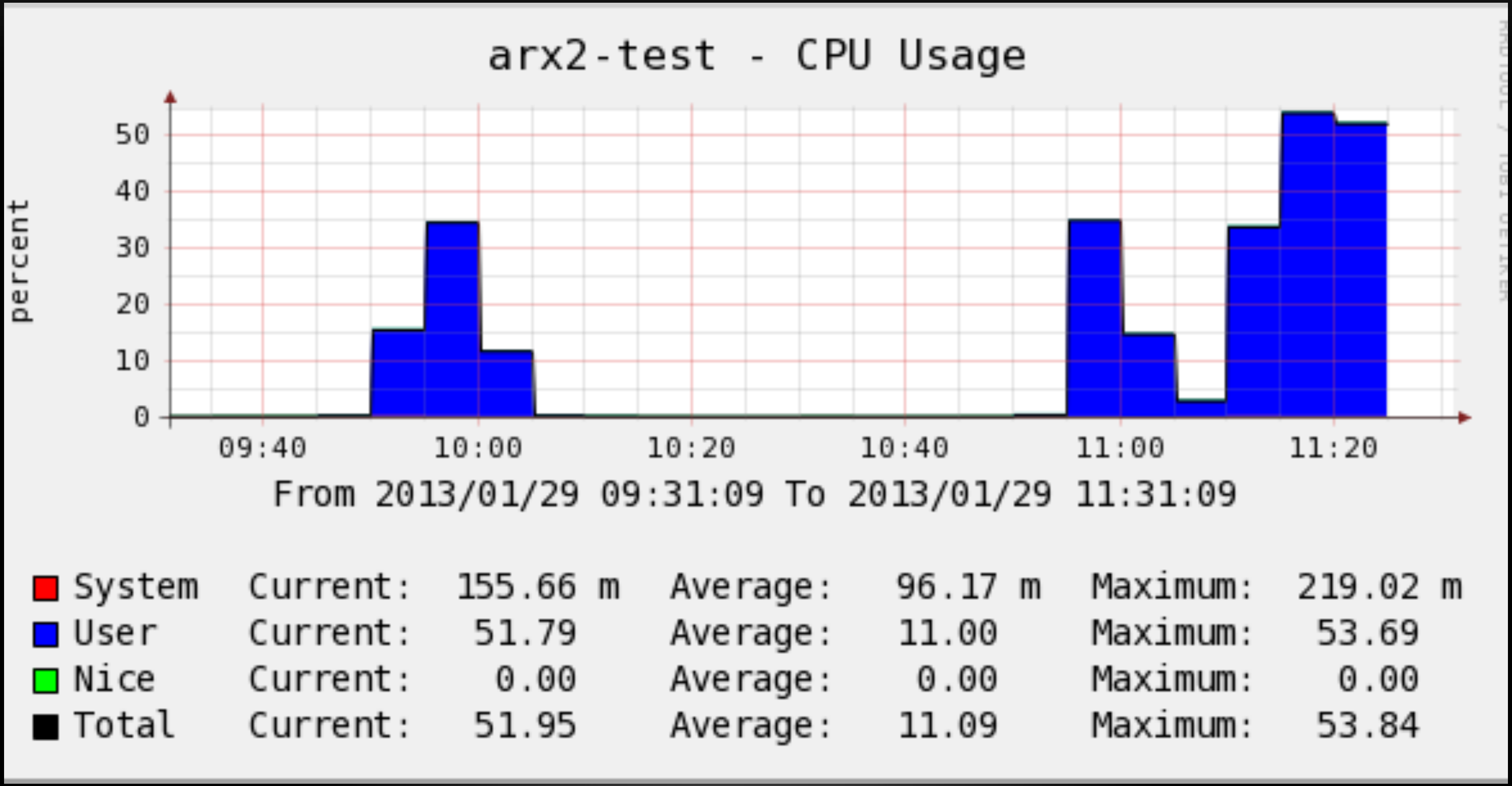
WORKING WITH OPENBGPD

The workaround…

- **Putting the peers into group's filter**
  - IPv4 peers
  - IPv6 peers
- **IPv4 prefix aggregation**
  - Huge number of prefix filter reduction

WORKING WITH OPENBGPD

Reduced routing convergence time…



BIRD FOR NEWBIE

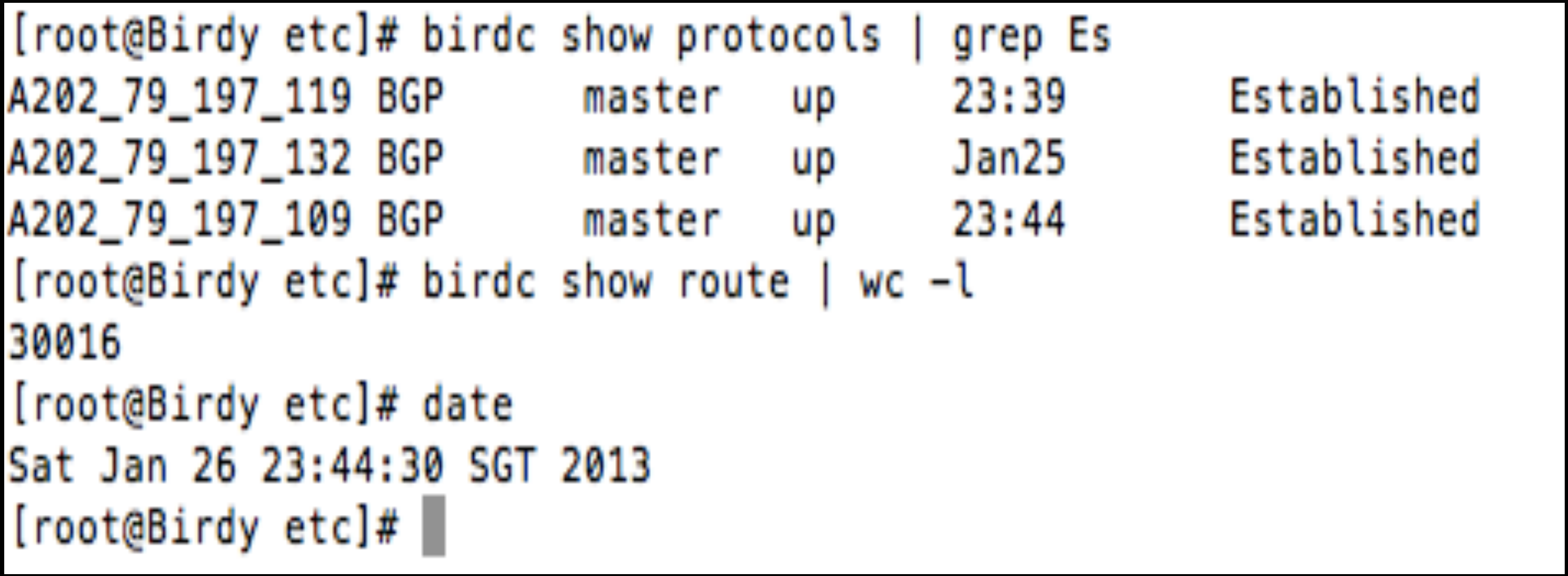
- **One process handles all BGP functions**
  - Separate instances for IPv4 and IPv6 though
- **Separate config files for IPv4 and IPv6**
- **BIRD BGP commands**
  - ‘birdc’ for IPv4 and ‘birdc6’ for IPv6
  - 2 ways to execute
    - Inside the ‘birdc’ mode
    - Outside the ‘birdc’ mode – more flexible

PLAYING WITH BIRD

- **Good in handling prefix filter**
  - Very fast routing convergence
- **Strict configuration change**
  - Change in some related neighbor parameters will flap the BGP session
    - Neighbor name – ‘protocol name’
    - Prepend flag
- **Strict execution of BGP commands**
  - Unable to do short form on the commands while executing outside ‘birdc’ mode

PLAYING WITH BIRD

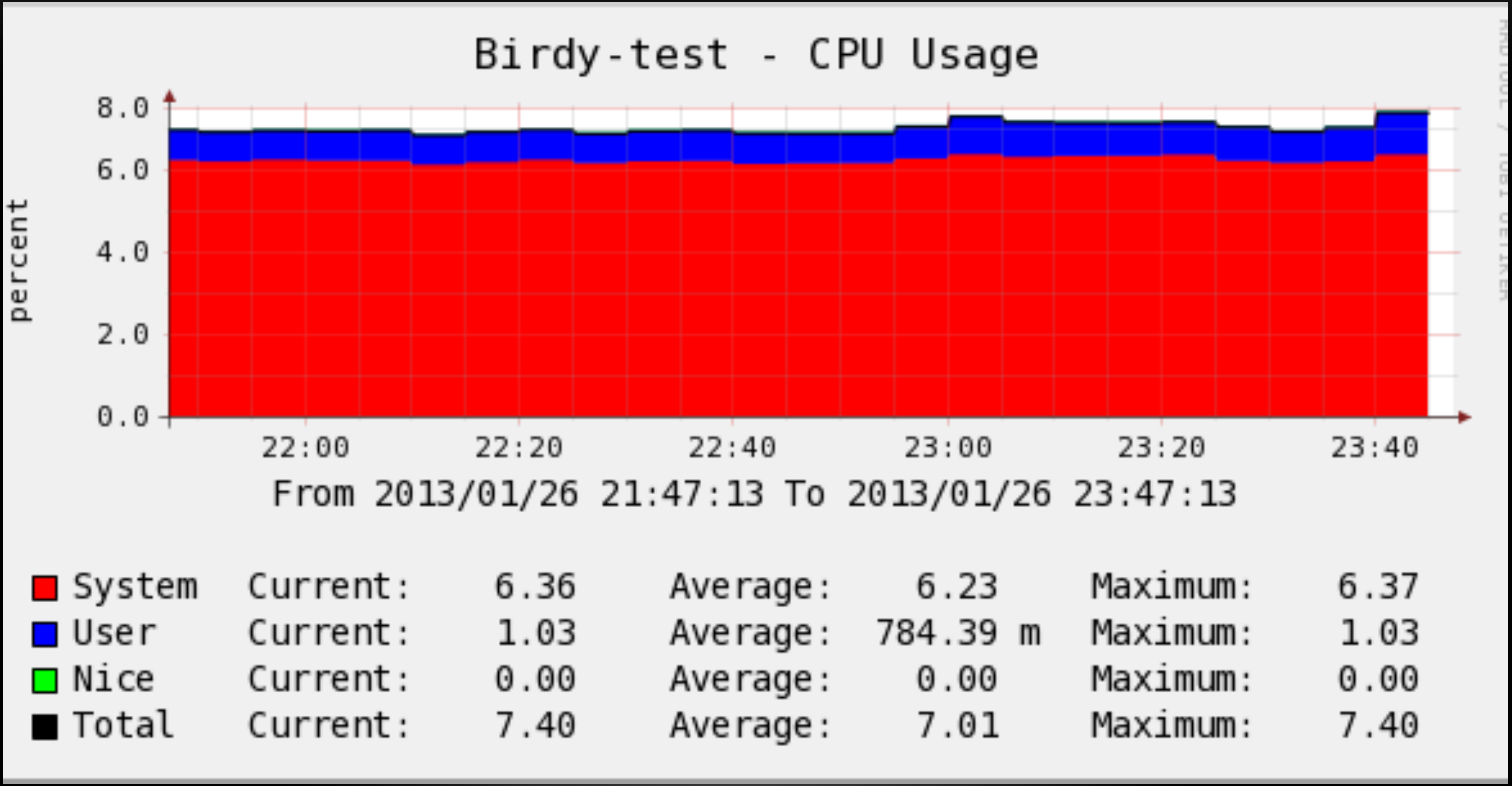
Very short routing convergence time…



```
[root@Birdy etc]# birdc show protocols | grep Es
A202_79_197_119 BGP      master   up     23:39       Established
A202_79_197_132 BGP      master   up     Jan25       Established
A202_79_197_109 BGP      master   up     23:44       Established
[root@Birdy etc]# birdc show route | wc -l
30016
[root@Birdy etc]# date
Sat Jan 26 23:44:30 SGT 2013
[root@Birdy etc]# 
```

PLAYING WITH BIRD

Very low CPU usage…



🟥	System	Current:	6.36	Average:	6.23	Maximum:	6.37
🟦	User	Current:	1.03	Average:	784.39 m	Maximum:	1.03
🟩	Nice	Current:	0.00	Average:	0.00	Maximum:	0.00
■	Total	Current:	7.40	Average:	7.01	Maximum:	7.40

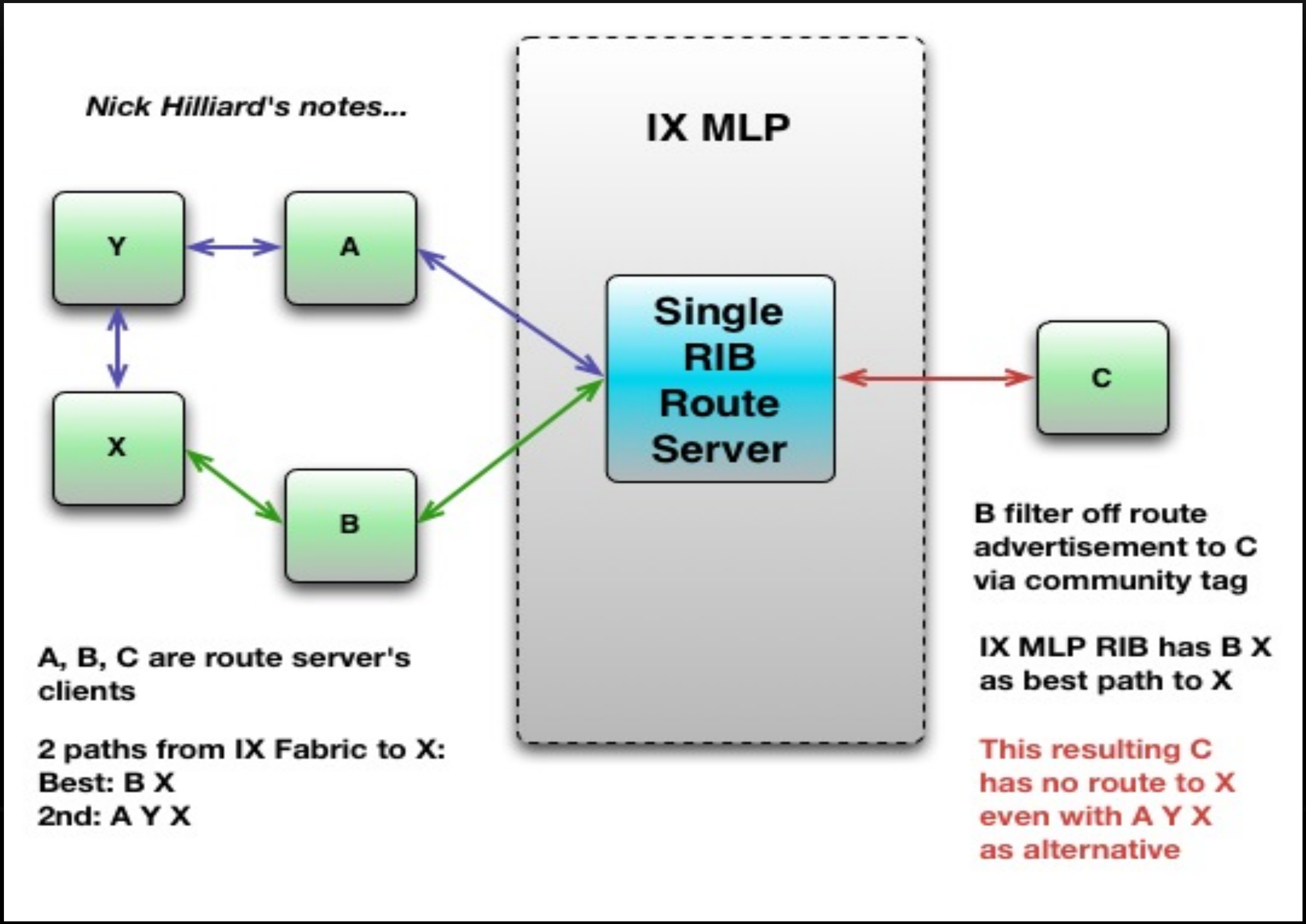
PLAYING WITH BIRD

The bad features…

- **No BGP uptime timer**
  - The uptime timer displayed is the uptime timer of the related protocol name
  - Soft BGP reload out will reset the protocol name’s uptime timer!!
  - Requested BIRD developers to include BGP uptime timer
- **No equivalent BGP “received-routes” command**
  - From my understanding, no way to get the routes that neighbor advertising before the filter
  - Still can see the routes that are advertised by neighbor and permitted by the filter

PLAYING WITH BIRD

Single RIB Problem Revisit…



Nick Hilliard's notes...

A, B, C are route server's clients

2 paths from IX Fabric to X:
Best: B X
2nd: A Y X

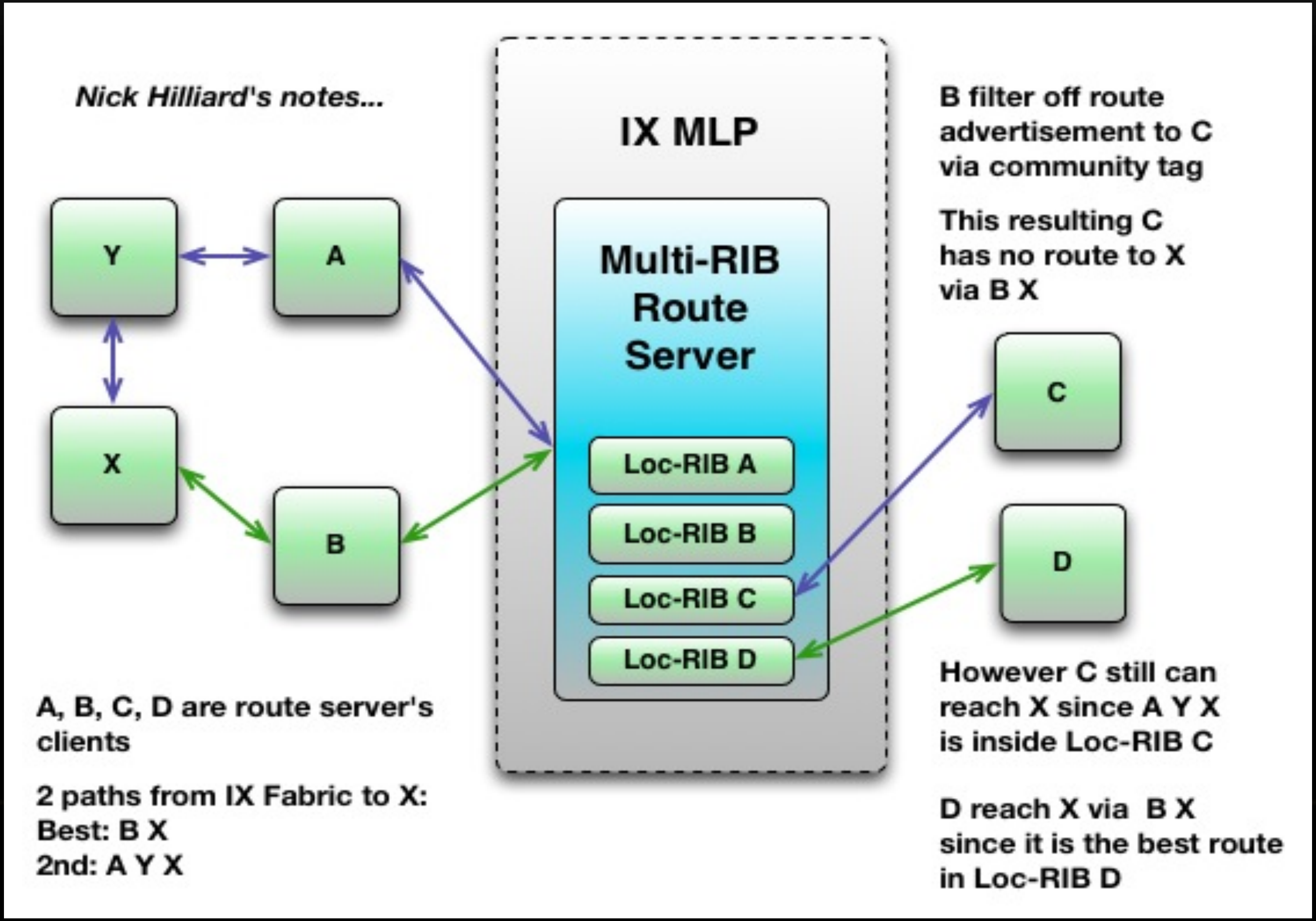
B filter off route advertisement to C via community tag

IX MLP RIB has B X as best path to X

This resulting C has no route to X even with A Y X as alternative

PLAYING WITH BIRD

Per-Client Loc-RIBs Revisit – Solution to Single RIB Problem



Nick Hilliard's notes...

B filter off route advertisement to C via community tag

This resulting C has no route to X via B X

However C still can reach X since A Y X is inside Loc-RIB C

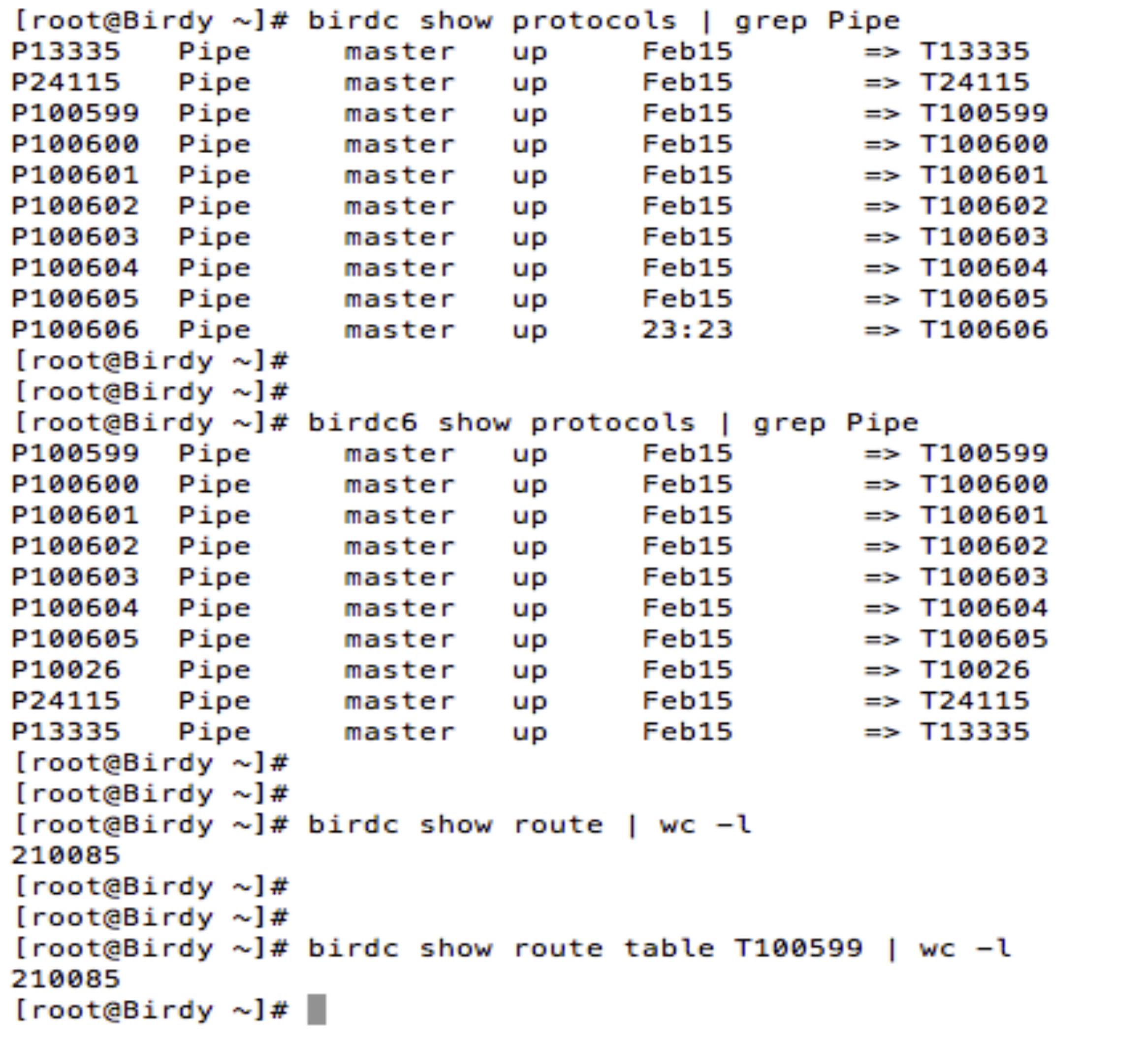
D reach X via B X since it is the best route in Loc-RIB D

A, B, C, D are route server's clients

2 paths from IX Fabric to X:
Best: B X
2nd: A Y X

PLAYING WITH BIRD…

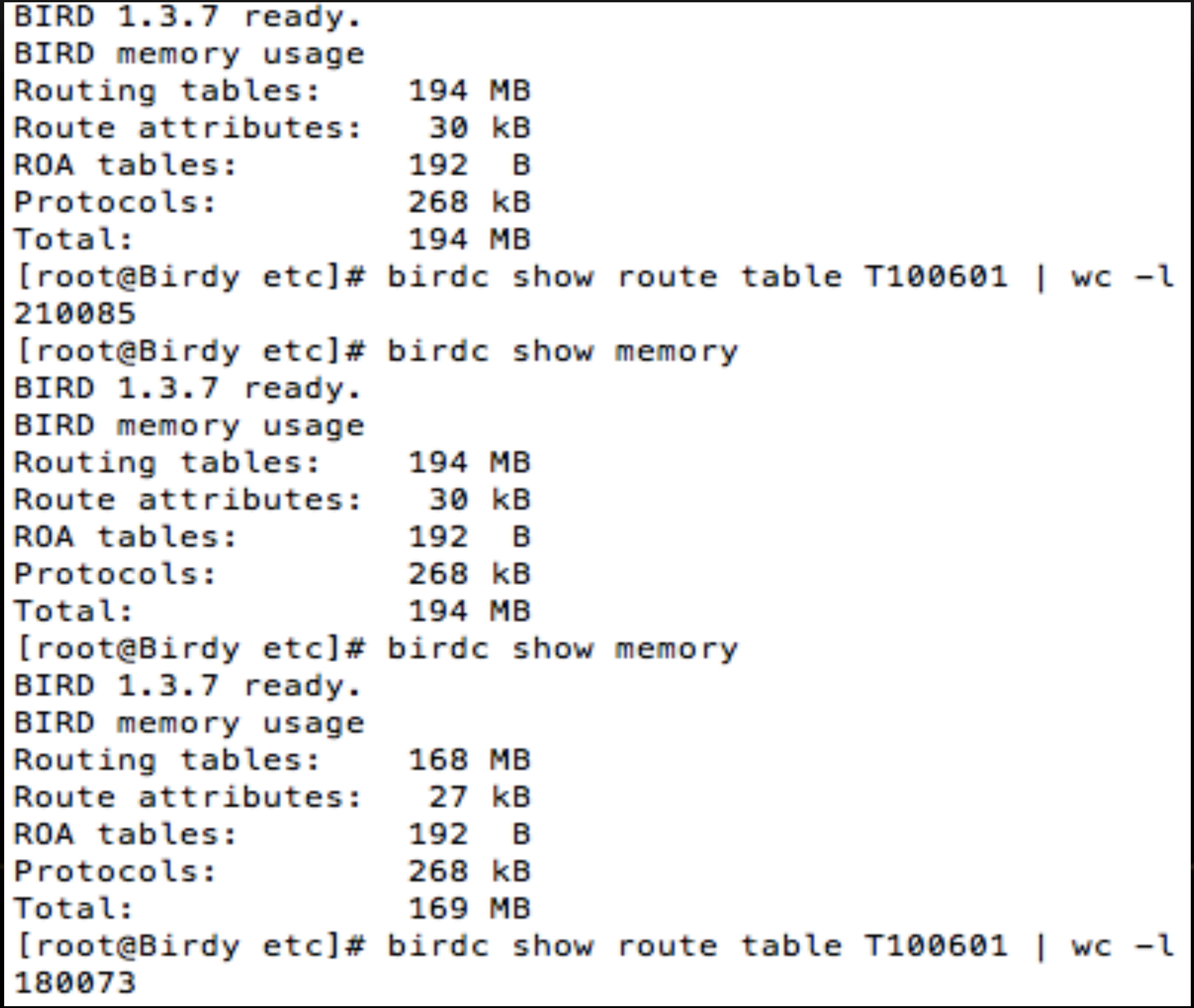
Testing Per-Client Loc-RIBs – 210K routes with 20 Loc-RIBs…



```
[root@Birdy ~]# birdc show protocols | grep Pipe
P13335    Pipe     master   up     Feb15        => T13335
P24115    Pipe     master   up     Feb15        => T24115
P100599   Pipe     master   up     Feb15        => T100599
P100600   Pipe     master   up     Feb15        => T100600
P100601   Pipe     master   up     Feb15        => T100601
P100602   Pipe     master   up     Feb15        => T100602
P100603   Pipe     master   up     Feb15        => T100603
P100604   Pipe     master   up     Feb15        => T100604
P100605   Pipe     master   up     Feb15        => T100605
P100606   Pipe     master   up     23:23        => T100606
[root@Birdy ~]#
[root@Birdy ~]#
[root@Birdy ~]# birdc6 show protocols | grep Pipe
P100599   Pipe     master   up     Feb15        => T100599
P100600   Pipe     master   up     Feb15        => T100600
P100601   Pipe     master   up     Feb15        => T100601
P100602   Pipe     master   up     Feb15        => T100602
P100603   Pipe     master   up     Feb15        => T100603
P100604   Pipe     master   up     Feb15        => T100604
P100605   Pipe     master   up     Feb15        => T100605
P10026    Pipe     master   up     Feb15        => T10026
P24115    Pipe     master   up     Feb15        => T24115
P13335    Pipe     master   up     Feb15        => T13335
[root@Birdy ~]#
[root@Birdy ~]#
[root@Birdy ~]# birdc show route | wc -l
210085
[root@Birdy ~]#
[root@Birdy ~]#
[root@Birdy ~]# birdc show route table T100599 | wc -l
210085
[root@Birdy ~]#
```

PLAYING WITH BIRD…

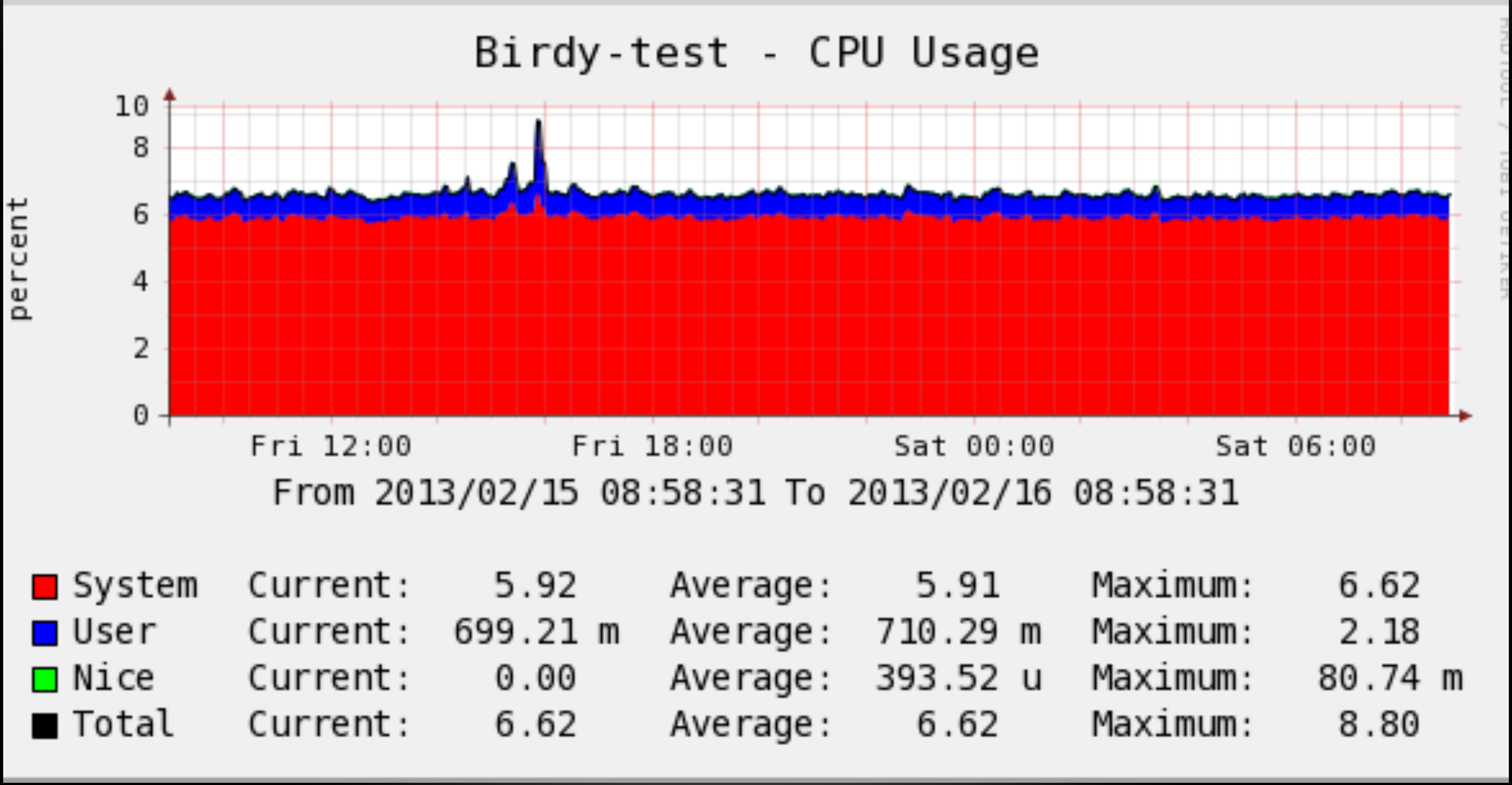
Testing Per-Client Loc-RIBs – 210K routes with 20 Loc-RIBs…



```
BIRD 1.3.7 ready.
BIRD memory usage
Routing tables:     194 MB
Route attributes:    30 kB
ROA tables:         192  B
Protocols:          268 kB
Total:              194 MB
[root@Birdy etc]# birdc show route table T100601 | wc -l
210085
[root@Birdy etc]# birdc show memory
BIRD 1.3.7 ready.
BIRD memory usage
Routing tables:     194 MB
Route attributes:    30 kB
ROA tables:         192  B
Protocols:          268 kB
Total:              194 MB
[root@Birdy etc]# birdc show memory
BIRD 1.3.7 ready.
BIRD memory usage
Routing tables:     168 MB
Route attributes:    27 kB
ROA tables:         192  B
Protocols:          268 kB
Total:              169 MB
[root@Birdy etc]# birdc show route table T100601 | wc -l
180073
```

PLAYING WITH BIRD

Testing Per-Client Loc-RIBs – CPU looks ok..

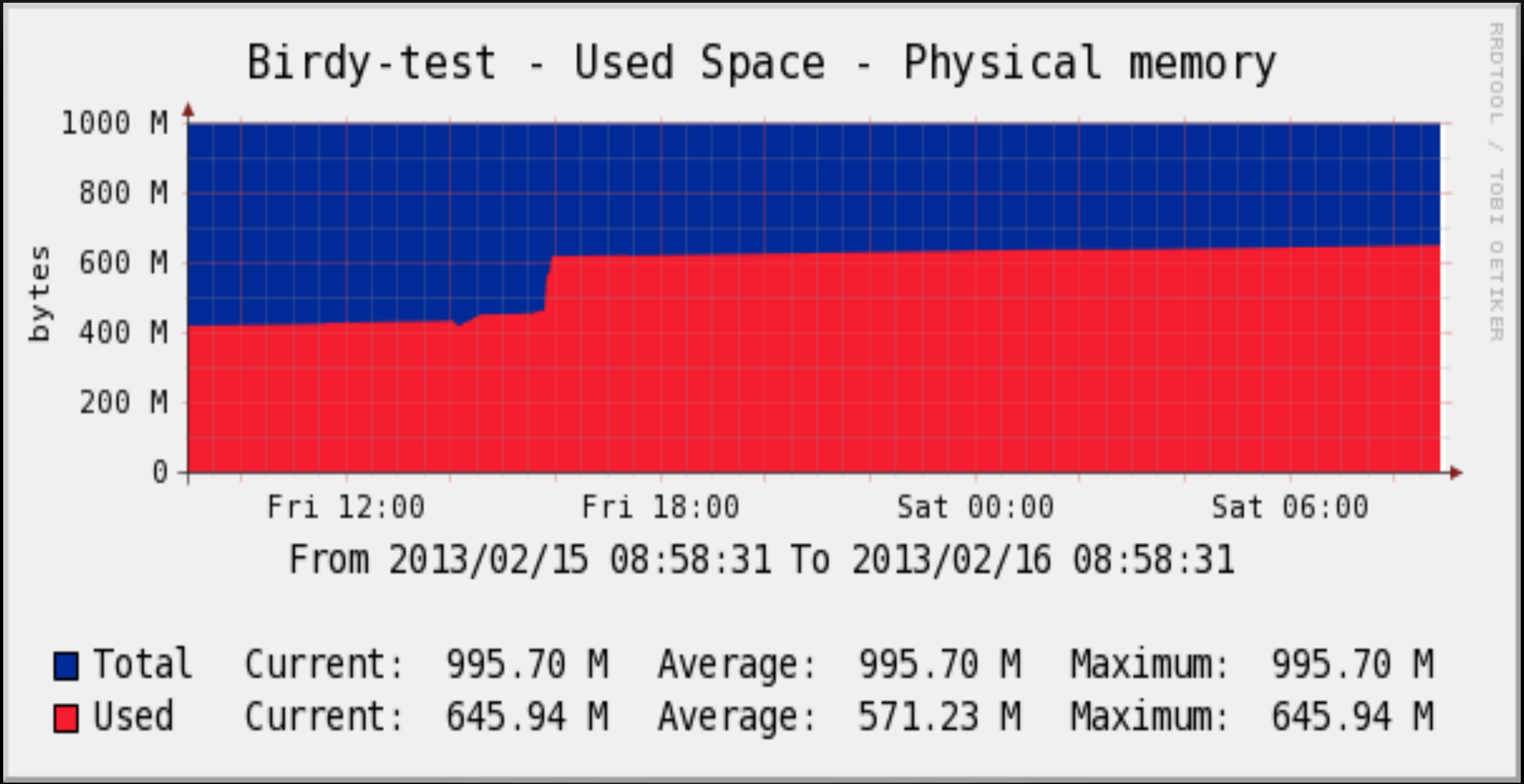


From 2013/02/15 08:58:31 To 2013/02/16 08:58:31

🟥 System	Current:	5.92	Average:	5.91	Maximum:	6.62
🟦 User	Current:	699.21 m	Average:	710.29 m	Maximum:	2.18
🟩 Nice	Current:	0.00	Average:	393.52 u	Maximum:	80.74 m
⬛ Total	Current:	6.62	Average:	6.62	Maximum:	8.80

PLAYING WITH BIRD

Testing Per-Client Loc-RIBs – Memory is depleting!!



PLAYING WITH BIRD

Testing Per-Client Loc-RIBs Summary...

- **Addresses the problem of Single RIB**
  - Single RIB's filter match best route only
  - Alternative routes will still be advertised in Per-Client Loc-RIBs scenario if the best route is filtered out
- **Increase in memory and CPU consumption**
  - The calculation changed from number of prefixes to number of clients and prefixes
- **Testing is still on going!!**
  - Need to ensure the performance of the route server will not be impacted with the implementation of Per-Client Loc-RIBs

OPENBGPD VS BIRD

OPENBGPD

- Three separate processes: parent, session engine, route decision engine
- Same config file for IPv4 and IPv6
- More flexible in executing BGP commands
- Flexible in doing configuration change
- No good in handling prefix filter resulting in long route convergence

BIRD

- One process but separate instances for IPv4 and IPv6
- Separate config files for IPv4 and IPv6
- More rigid in executing BGP commands
- Strict configuration change
- Good in handling prefix filter resulting in very short route convergence

WHAT IS THE CHOICE?

- **BIRD to go for?**
- **Software bugs**
  - Get the stable version
  - Dual routing daemon's approach?
- **Keep on testing!!**

THANK
YOU