

Traffic Analysis of UDP-based Flows in ourmon

Jim Binkley, Divya Parekh
jrb@cs.pdx.edu, divyap@pdx.edu
Portland State University
Computer Science
Courtesy of John McHugh

Outline

- ❑ problem space - and short ourmon intro
- ❑ UDP flow tuple
 - UDP work weight
 - UDP guesstimater
 - problems (DNS and p2p as scanners)
- ❑ packet-size based UDP application guessing
- ❑ conclusions

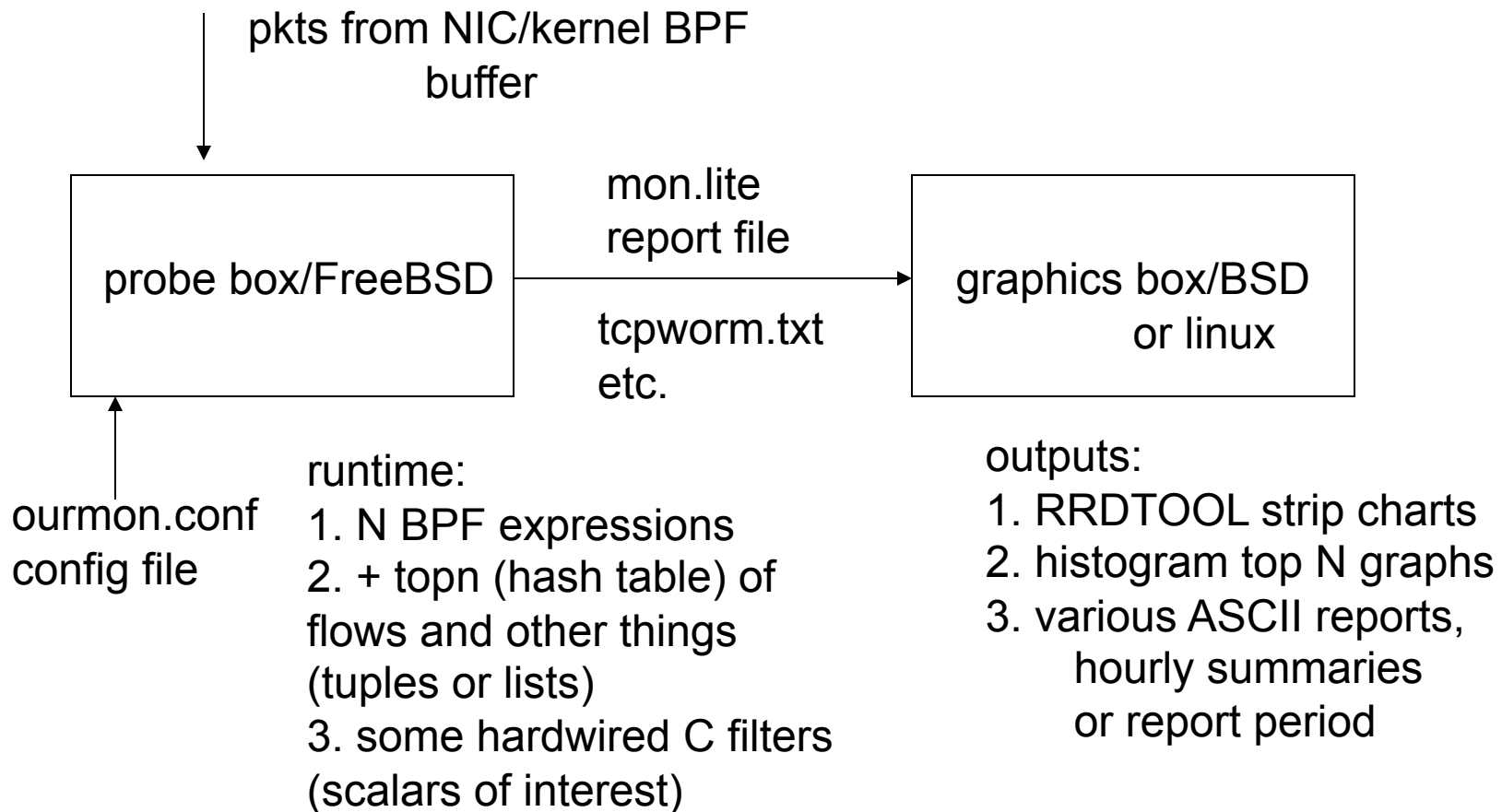
motivation - problem space

- ❑ UDP-based DOS attacks certainly exist
- ❑ p2p searching courtesy of Distributed Hash Tables on the rise (use UDP to search and TCP to fetch)
 - Kademia protocol - Maymounkov and D. Mazieres.
- ❑ stormworm botnet is UDP/P2P based
 - based on edonkey related protocol (overnet)
- ❑ p2p-based apps not just for file-sharing
 - Joost - “cable TV”, Skype - VOIP
- ❑ goal: focus on UDP flow activity in terms of security and p2p

brief ourmon intro

- ❑ 2 part system: front-end, back-end
 - front-end: packet sniffer, output ASCII files
 - back-end: web-interface with graphs, and aggregated logs
- ❑ front-end produces:
 - scalars that produce RRDTOOL web graphs
 - either hardwired or programmable (BPF)
 - various kinds of top-N lists (ourmon flows)
- ❑ back-end
 - web access plus graphics processing, log aggregation
 - 30-second view and hourly aggregation views
 - event log for important security events

ourmon architectural breakdown



filters: BPF expressions, lists, some hardwired C filters

ourmon flow breakdown

- top N traditional (IP.port->IP.port) flows
 - IP, UDP, TCP, ICMP
 - hourly summarizations and web histograms
- IP host centric flows at Layer 4
 - TCP (presented in TCP port report)
 - **UDP (presented in UDP port report) <-----**
(this is what we are talking about here)
- Layer 7 specific flows now include
 - IRC channels and hosts in channels
 - DNS and ssh flows (spin-off of traditional flows)

UDP port report

- ❑ UDP centric top N tuple collected by front-end every 30 seconds
- ❑ hourly summarizations made by back-end
- ❑ flow tuple fields:
 - IP address - key
 - IP dst address - one sampled IP dst
 - UDP work weight - noise measurement (sort by)
 - SENT - packet count of packets sent
 - RECV - packet count of packets returned to IP
 - ICMPERRORS - icmp errors returned (unreachables in particular)

UDP port report tuple, cont.

- ❑ L3D - count of unique remote IP addresses in 30-second sample period
- ❑ L4D - count of unique remote UDP dst ports
- ❑ SIZEINFO - size histogram
 - 5 buckets, $\leq 40, 90, 200, 1000, 1500$
 - (this is L7 payload size)
- ❑ SA - running average of sent payload size
- ❑ RA - running average of recv. payload size
- ❑ APPFLAGS - tags based on L7 regular expressions
 - s for spim, d for DNS, b for Bittorrent, etc.
- ❑ PORTSIG - first ten dst ports seen with packet counts expressed as frequency in 30 sec report
 - e.g., [53,100] meaning 100% sent to port 53

UDP work weight calculation

- per IP host
- **UDP ww = (SENT * ICMPERRORS) + RECV**
 - if ICMPERRORS == 0, then just SENT + RECV
- we sort the top N report by the UDP ww
- basically can divide results up into about 3 bands: (numbers are relative to ethernet speed, 1 Gbit in our case)
 - TOO HIGH (> 10 million in our case)
 - BUSY 1000..1 million (p2p/games/dns servers)
 - LOW (most - e.g., clients doing DNS) < 1000

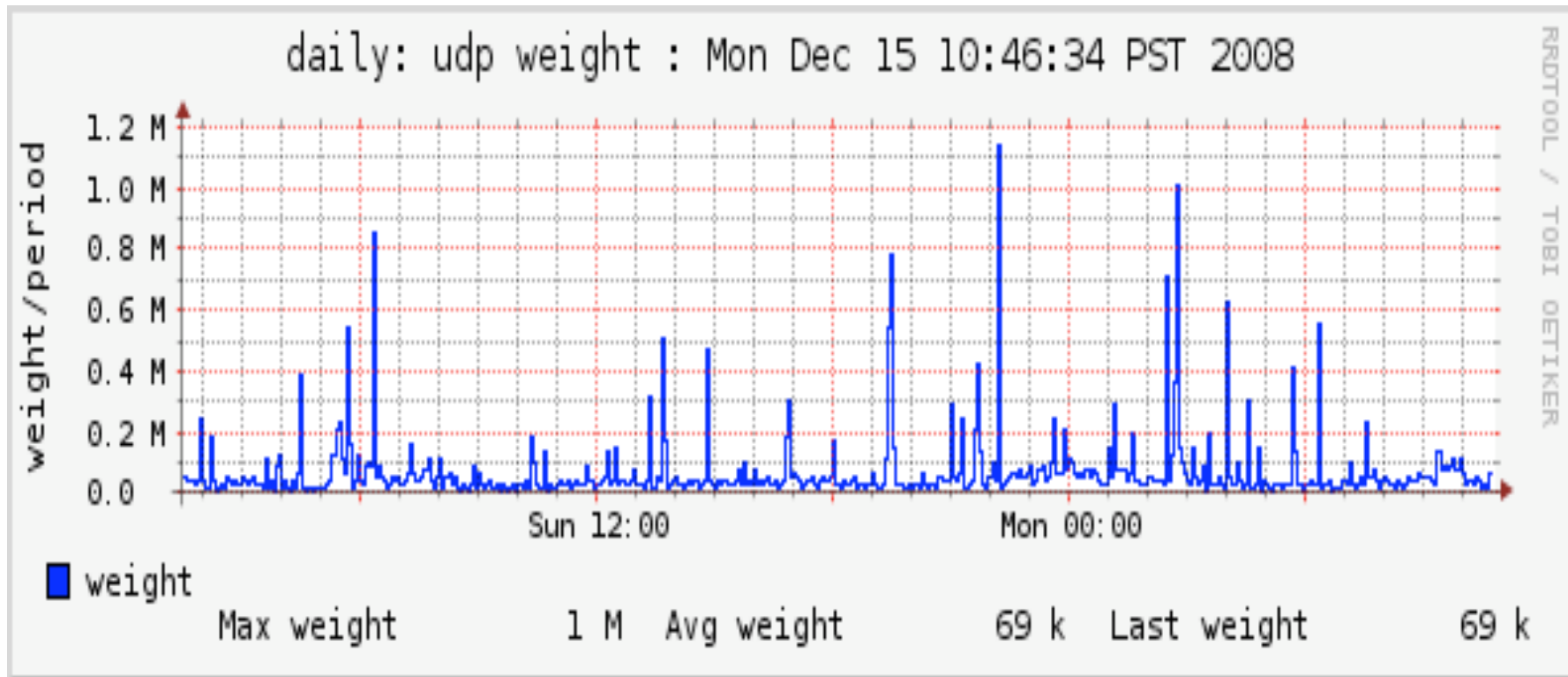
theory behind UDP workweight

- ❑ if a host is doing
 - scanning
 - p2p
- ❑ it may generate SENT * ERROR packets and hence appear higher in the report
- ❑ scanning error generation is obvious
- ❑ p2p error generation is because a p2p host has a set of peers, some of which are stale
- ❑ if just busy, we add SENT + RECV
 - some hosts may recv more packets than they send
 - e.g., JOOST p2p video apps
- ❑ result: big error makers to the top, busy hosts next

some added features of UDP work weight

- we graph the very first tuple (the winner!) over the day, which
 - gives an average distribution
 - shows spikes
 - average day shown in next slide
- if work weight > HIGH THRESHOLD
 - we record N packets with automated tcpdump mechanism
 - this has proved effective at the past in catching DOS attacks sources and targets
 - even when monitoring fails if DOS was too much for probe - so far have always managed to capture sufficient packets

daily graph of top UDP work weights



top single work weight per 30-second period for typical day:
note: peaks here are usually SPIM outside in

contrived UDP port report (simplified)

IP src	ww	Guess	SENT	RECV	ICMP ERR	L3D / L4D	App flags	portsig
1*	20 million	scan	20000	18000	827	208 / 527	b	many
2	12 million	ipscan	6598	12	1936	600 / 2	s	1026, 1027
3*	49000	p2p	1555	1215	31	1637 / 1297	b	many
4	3321	p2p	2430	891	1	703 / 279	d	53

UDP guesstimater algorithm

- ❑ attempt to guess what host is up to based on attributes
- ❑ principally on L3D/L4D and workweight
- ❑ goal: use only L3 and L4 attributes not L7 attributes and avoid destination port semantics
 - thus it should work if bittorrent is on port 53 and encrypted
- ❑ per IP host guess
- ❑ basically a decision tree with 3 thresholds
 - WW high threshold - set at 10 million
 - L3D/L4D - p2p counts (say 10 for a low threshold)

rough algorithm

- ❑ guess = “unknown”
- ❑ if $ww > \text{HIGHTHRESHOLD}$
 - guess = scanner
 - if L4D is HIGH and L3D is LOW
 - guess = portscanner
 - else if L3D is HIGH and L4D is LOW
 - guess = ipscanner
- ❑ else if L3D and L4D $> \text{P2PTHRESHOLD}$
 - guess = p2p
- ❑ we have HIGHTHRESHOLD at 10million, port thresholds at 10 (might be higher/lower depending on locality)

how well does it work?

- ❑ it is really only pointing out obvious attribute aspects but this is helpful to a busy analyst
- ❑ two interesting errors
- ❑ 1. because DNS servers are typically busy and because they send to many ports, many destinations
 - diagnosed as p2p -- true, but somehow annoying
 - our L7 pattern is complex and is probably sufficient as DNS isn't going to be encrypted
- ❑ 2. some p2p hosts -- typically with stale caches may be diagnosed as “scanners”
 - in a sense this is true
 - note that p2p/scanner overlap is a long-standing problem

application guessing - limited experiment

- ❑ inspired by Collins, Reiter: Finding Peer-To-Peer File Sharing Using Coarse Network Behaviors, Sept. 2006
- ❑ decided to try to use packet sizes to see if we could guess UDP-based applications
- ❑ SIZEINFO SA/RA fields used for the most part
 - thus 7 attributes in all, basic sent size histogram + SA,RA
- ❑ initially only done if guesstimater guesses “p2p”
 - had to back that off for Skype
- ❑ only tested in a lab using Windows Vista and applications (some testing on a MAC)
- ❑ culled stats from 30 second UDP port reports
- ❑ this information is appended to guess e.g.,
 - p2p:joost

approach

- ❑ limited testing - lab only (barring stormworm where we got pcap traces from elsewhere)
- ❑ gathered attribute stats and
 - graphed them
 - per attribute choose lower and upper threshold based on $\geq 90\%$ of samples
 - note that the 1000-1500 byte SIZE attribute was always 0 (not used)
- ❑ result coded as decision tree forest
 - really a set of if tests - not if-then-else
 - therefore results could overlap (fuzzy match)

apps/protocols in experiment

application	protocol
edonkey	emule
bittorrent	bittorrent
azureus	bittorrent
utorrent	bittorrent
limewire	gnutella or bittorrent
joost	joost
skype	skype
stormworm (UDP)	emule variant

results?!

- ❑ suggestive and interesting but not 100% conclusive that this approach might be valuable
- ❑ problems:
 - not enough testing but seemingly worked well barring skype
 - not enough apps (should have included DNS! and probably NTP)
 - we may be finding app classes not particular apps
 - we don't know all the p2p apps on our network
 - it is a university, although bittorrent and gnutella are dominant
 - perhaps should have more buckets, look at recv packet buckets. better threshold estimation, etc.
 - we could not get skype to behave - could catch it sometimes, other times not, not necessarily p2p, not necessarily UDP

conclusions

- ❑ UDP centric port tuple is useful for host behavior analysis
 - with simple stats and a top N sort
- ❑ UDP ww is a good simple stat
 - helps up track down blatant security problems
 - measure of noise and load
- ❑ guesstimater is useful in terms of
 - dividing world into security threats vs p2p based on non-L7 data
 - saving time spent looking at data
 - best to learn DNS servers though
- ❑ application guessing
 - promising -- would be nice if researchers elsewhere would pursue it as well

ourmon on sourceforge

- ❑ open source
- ❑ new release (2.9) including work here expected Spring 2009
 - UDP port report guesstimator etc, plus hourly UDP summarization for port report
 - ssh flow statistics (global site logging)
 - expanded DNS statistics (errors, top N queries)
 - expanded blacklist mechanism (can handle net/mask)
- ❑ ourmon.sourceforge.net (version 2.81)
 - currently supports threads in front-end