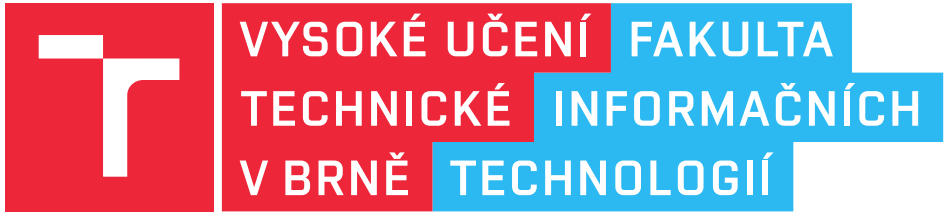
Akcelerace aplikace pro potlačení DDoS útoků

Andrei Shchapaniak

Vedoucí: Ing. Jan Kučera

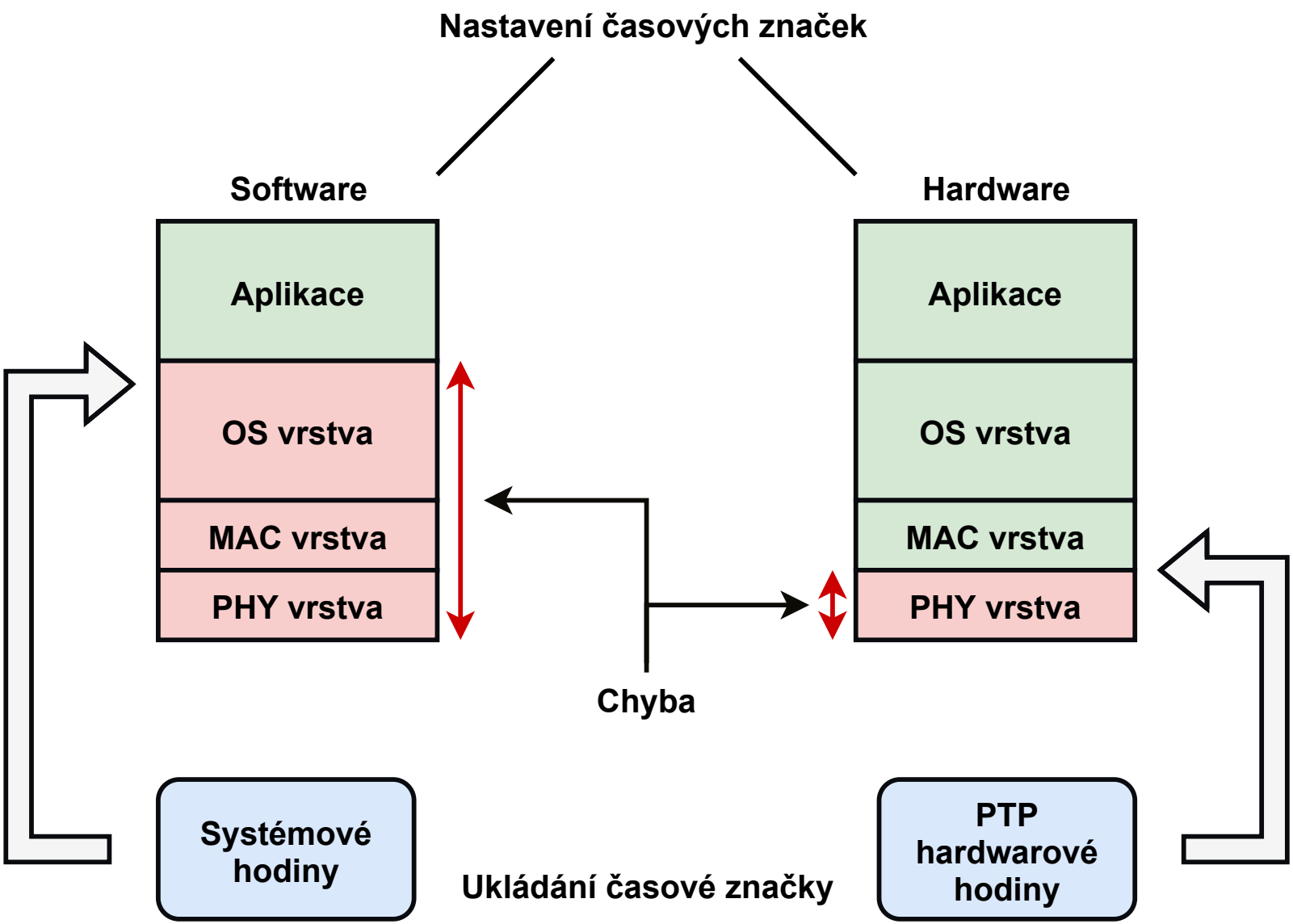


28. ledna 2022

Obsah

- Podpora offloadu časových značek
- Spouštění služby čističky pod neprivilegovaným uživatelem/skupinou
- Přidání měření code coverage pro unit testy

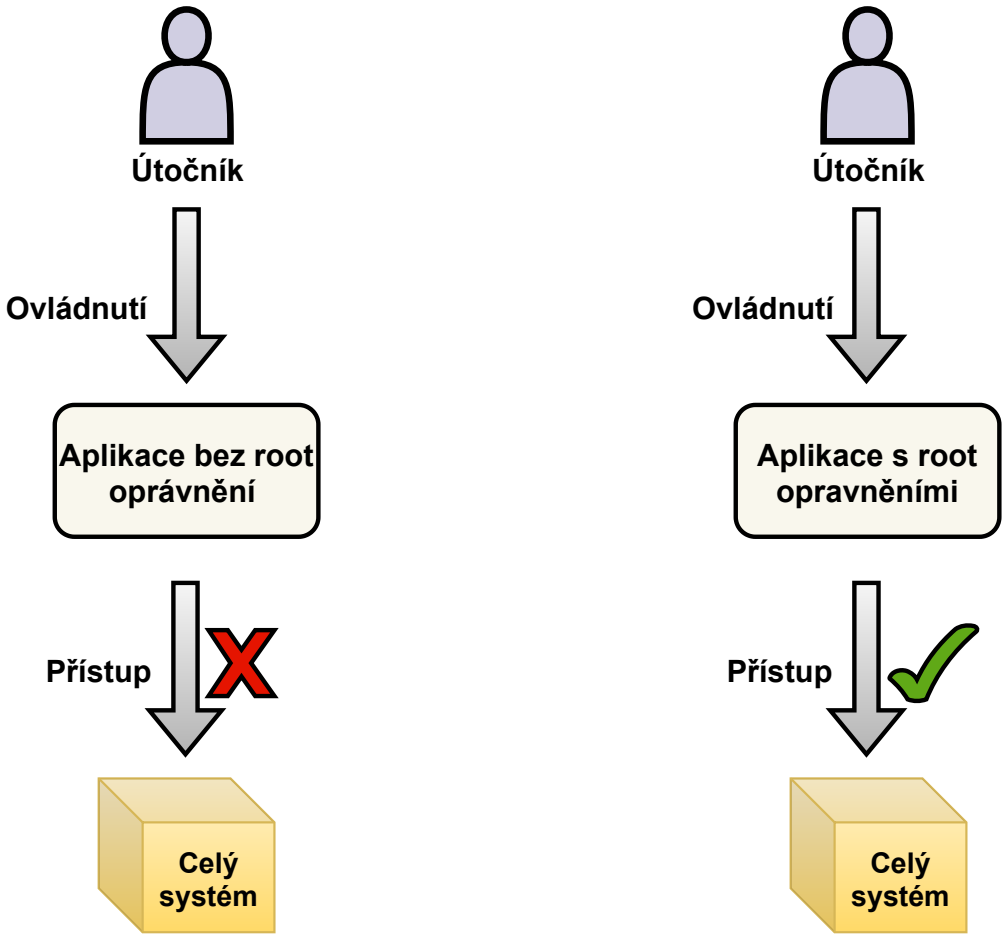
Nastavení časových značek



Shrnutí výsledků pro podporu časových značek:

Název karty	Výsledky
Mellanox Connect-X6	Je možnost pomocí mlxconfig tooly nakonfigurovat formát značek přímo na reálný čas, avšak ze strany DPDK není možnost ověřit, že je karta takto nakonfigurována
Mellanox Connect-X5/X6	Je možnost časovou značku na reálný čas převést voláním funkce mlx5dv_ts_to_ns(), kterou obsahuje knihovna, která není skrz DPDK jednoduše dostupná.

K čemu je vhodná možnost spouštění aplikace bez root?

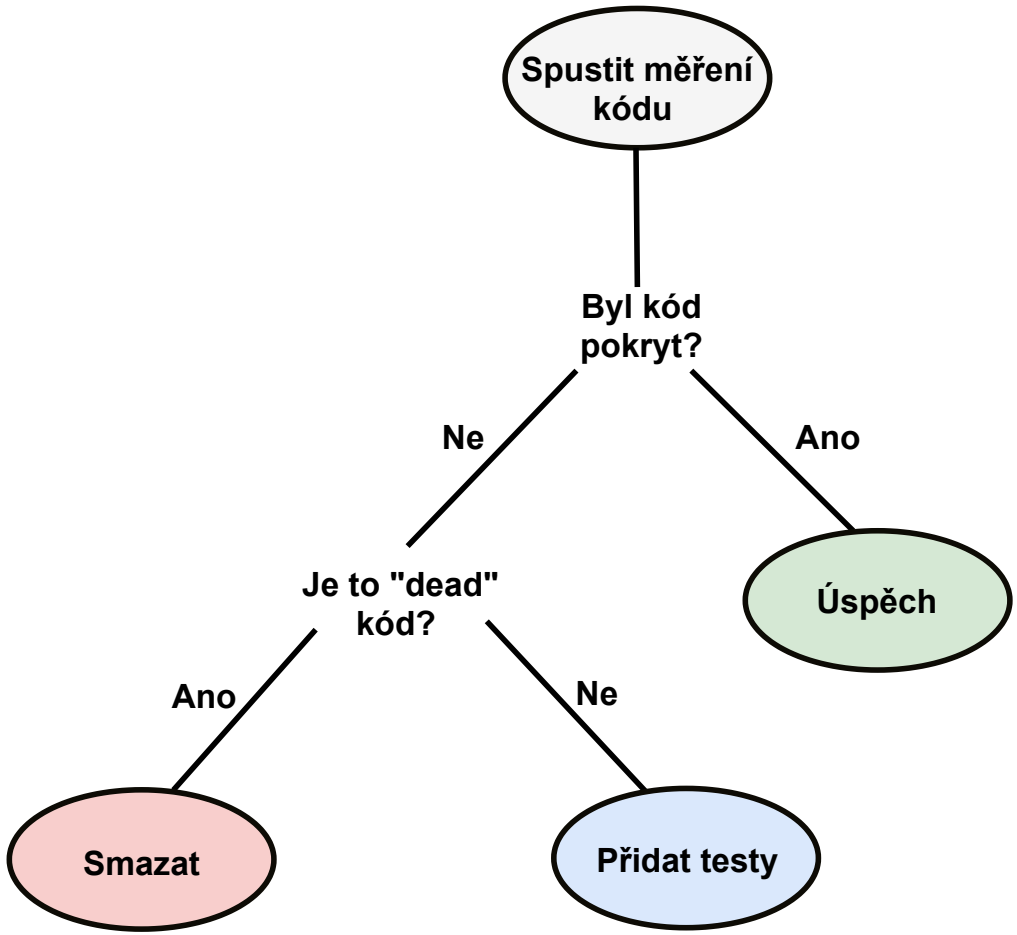


Dosažené výsledky

- Rozšíření připojování velkých stránek. Uživateli je poskutnuta možnost zvolit si jméno uživatele a skupiny, pro které budou umožněny manipulace s velkými stránkami čističkou
- Byl přidán volitelný parametr pro místo alokace velkých stránek
- Zvyšená oprávnění mají pouze speciální soubory, ke kterým přistupuje čistička
- RPM balíček zajišťuje instalaci nutných komponent a jejich nastavení

Code Coverage ukazuje jak velká část kódu je spouštěna, když běží automatické testy.

- Hledání kusů kódu, které jsou/nejsou pokryty testy
- Smazání tzv. dead kódu



Statistika pokrytí kódu

File	Lines			Branches	
abstract_intel_rss_configurer.c		0.0%	0 / 232	0.0%	0 / 58
abstract_mempool_manager.c		0.0%	0 / 65	0.0%	0 / 18
abstract_port_configurer.c		0.0%	0 / 276	0.0%	0 / 108
all_lcore_pool.c		3.8%	5 / 131	0.0%	0 / 73
anything.h		0.0%	0 / 2	-%	0 / 0
anything_container.c		28.6%	2 / 7	0.0%	0 / 2
appfs.c		36.4%	4 / 11	0.0%	0 / 2
appfs_aggregator.c		4.2%	12 / 286	0.0%	0 / 151
appfs_fuse.c		7.0%	21 / 299	5.5%	6 / 110
appfs_holder.c		9.3%	13 / 140	3.9%	2 / 51
appfs_notifier.c		11.1%	7 / 63	0.0%	0 / 14
appfs_point.c		8.3%	8 / 96	0.0%	0 / 52
appfs_root.c		63.9%	152 / 238	32.6%	43 / 132
appfs_root.h		33.3%	6 / 18	20.0%	2 / 10
appfs_string_interpolator.c		5.7%	4 / 70	0.0%	0 / 24
application.c		8.7%	6 / 69	0.0%	0 / 44
application_module.h		0.0%	0 / 15	0.0%	0 / 12
async_executor.c		3.5%	8 / 229	0.0%	0 / 50
basic_pkt_ip_filter.c		37.5%	131 / 349	23.1%	31 / 134
basic_pkt_proc_runner.c		6.3%	13 / 205	0.0%	0 / 51
dashboard_panel.c		0.0%	0 / 3	0.0%	0 / 2
dashboard_shell.c		6.6%	5 / 76	0.0%	0 / 36
dcpro_ethdev_dump.c		2.7%	5 / 188	0.0%	0 / 69
direct_port_id.c		12.5%	5 / 40	0.0%	0 / 14
direct_sys_if_namer.c		30.0%	3 / 10	0.0%	0 / 4
dpdk_appfs.c		1.7%	4 / 233	0.0%	0 / 74
dynarray.c		77.0%	67 / 87	67.3%	35 / 52
dynarray.h		77.8%	7 / 9	51.3%	20 / 39
ext_ring_keeper.c		13.6%	8 / 59	0.0%	0 / 24
file_sql_query_manager.c		4.3%	5 / 115	0.0%	0 / 62
generic_mempool_manager.c		12.6%	13 / 103	0.0%	0 / 26
generic_netlink_gatherer.c		7.1%	11 / 156	0.0%	0 / 76
generic_port_configurer.c		23.4%	22 / 94	0.0%	0 / 8
host_traffic.c		52.4%	33 / 63	36.4%	16 / 44
host_traffic_provider.c		3.2%	6 / 186	0.0%	0 / 54
htab_imm_bucket.c		94.4%	17 / 18	50.0%	2 / 4

Rozkliknutý soubor

43		8	`static inline uint16_t rte_pktmbuf_alloc_burst(`
44			`struct rte_mempool *pool,`
45			`struct rte_mbuf **mbufs,`
46			`uint16_t count)`
47			`{`
48		8	`uint16_t idx = 0;`
49		8	`int rc;`
50			
51	▸ 1/2	8	`rc = rte_mempool_get_bulk(pool, (void **) mbufs, count);`
52	▸ 1/2	8	`if (unlikely(rc)) {`
53			`struct rte_mbuf *m;`
54			
55		✗	`for(idx = 0; idx < count; idx++) {`
56		✗	`rc = rte_mempool_get(pool, (void **) &m);`
57		✗	`if (unlikely(rc))`
58			`break;`
59			
60		✗	`rte_pktmbuf_reset(m);`
61		✗	`mbufs[idx] = m;`
62			`}`
63			
64		✗	`return idx;`
65			`}`

Děkuji za pozornost!

Prostor pro Vaše dotazy.