Proiectarea şi implementarea middleboxurilor software practice

Sumar

Universitatea Polithenica Bucureti



Teză de doctorat

Vladimir Olteanu

Coordonator: Valentin Cristea

Cuvinte-cheie

- reţele
- datacenter
- middlebox
- protocol

Sumar

Această teză abordează proiectarea middlebox-urilor software scalabile orizontal.

Middlebox-urile sunt maşini care procesează pachete de reţea, efectuând operaţiuni care pot fi simple, ca filtrarea lor pe baza unor cămpuri, sau complexe, ca detectarea scurgerilor de informaţii secrete. Ele sunt un element de bază al internetului de astăzi.

În mod tradiţional, middleboxurile erau dispozitive hardware monolitice. Aducerea lor la zi, fie din punct de vedere al performanţei, fie al funcţionalităţii, era de obicei un lucru costisitor, deoarece însmena înlocuirea întregii maşini cu un model diferit. În plus, era necesar ca middleboxurile să facă faţa volumului de trafic de la orele de vârf; în afara acestor ore, capacitatea suplimentară era, în esenţă, irosită.

Pentru a rezolva aceste probleme, a apărut un nou domeniu de cercetare, numit Network Function Virtualization. Ideea de bază este că procesarea traficului nu se mai face in dispozitive dedicate, ci în software care rulează într-o manieră distribuită pe hardware generic. Astfel de middleboxuri pot fi scalate orizontal pe măsură ce volumul de trafic creşte sau scade: capacitate suplimentară poate fi adăugată pornind câteva mssini în plus, şi apoi eliminatâ oprind unele dintre ele.

Cel mai mare obstacol din calea scalabilităţii orizontale a meddieboxurilor este faptul că acestea pot ţine stare, atât specifică fiecărui flow în parte, cât şi partajatâ de mai multe flow-uri. Această stare este indispensabilă pentru procesarea corectă a traficului; dacă starea ar deveni indisponibilă, chiar şi temporar, toate flow-urile care îi corespund ar putea suferi întreruperi. Prin urmare, scalarea orizontală nu implică doar redirecţionarea unei părţi a traficului către o altă maşină, ci şi migrarea tuturor stărilor relevante către maşina menţionată.

S-au făcut eforturi pentru a găsi un framework general care să permită evenimente de scalare orizontală aproape fără întreruperi, dar acestea tind să impună un cost destul de mare în ceea ce priveşte performanţa. Subtextul acestei teze este că argumentăm împotriva unor astfel de soluţii generalizate; în acest scop, proiectăm două middleboxuri separate şi folosim soluţii diferite pentru a le rezolva problemele de scalabilitate.

Primul middlebox pe care îl discutăm este un carrier-grade NAT. Acesta este un middlebox destul de simplu de implementat, care ţine stare atăt pentru fiecare flow, cât şi pentru fircare IP extern. Implementarea noastră are un algoritm de migrare a stării care impune o penalizare minimă de performanţă. Aici arătăm că, pur şi simplu riscând reordonarea unor pachete, dar păstrând toate celelalte garanţii de funcţionare, perturbările de performanţă impuse de evenimentele de scalare sunt minime. Reordonările ocazionale ale pachetelor sunt, în general, un lucru căruia protocoalele de la nivelul 4 şi mai sus îi pot face faţă cu uşurinţă.

Al doilea middlebox discutat este Beamer, un load-balancer stateless pentru cloud, care funcţionează şi cu MPTCP. Load-balancer-ele stateful îşi amintesc asocierile dintre conexiuni şi serverele care se ocupă de ele. Aceasta, în esenţă, încearcă să dubleze informaţiile păstrate de serverele înseşi în tabelele lor de conexiuni. Prin faptul că nu ţine niciun fel de stare, Beamer este uşor de implementat atât în software, cât şi în hardware, şi evită multe dintre problemele cu care se confruntă load-balancer-ele in mod general. Beamer poate face faţă cu uşurinţă atacurilor de tip SYN flood, ceva cu care toate load-balancer-ele stateful au probleme.

În cele din urmâ, vom discuta despre ceva care este un middlebox doar în sensul larg al cuvântului: am proiectat versiunea 6 a protocolului SOCKS şi am implementat un proxy. Versiunea 6 tratează multe dintre problemele versiunii 5: penalizarea din punct de vedere al RTT-urilor este, în majoritatea cazurilor, zero (sau chiar negativă!), un serviciu DNS implicit este furnizat clienţilor, şi protocolul profită din plin de noile îmbunătăţiri introduse în TLS 1.3. Deoarece, strict vorbind, proxy-ul este un serviciu care rulează peste TCP, deployment-uri mari de SOCKSv6 pot fi făcute folosind Beamer.

Cuprins

1 Introducere **3**

2 Fundal **5**
2.1 Middleboxurile în arhitectura Internetului 5
2.2 OpenFlow şi SDN 6
2.3 Multipath TCP 7

3 Principii pentru procesarea scalabilă de pachete **9**
3.1 Starea de fapt 12
3.2 Contribuţiile tezei 13
3.3 Publicaţii 16

4 Carrier-grade NAT **17**
4.1 Introducere şi motivare 17
4.2 Proiectarea unui Carrier-Grade NAT 19
4.3 Implementare 22
4.4 Evaluare 24
4.5 Concluzii 29

5 Beamer **31**
5.1 Introducere 31
5.2 Fundal 32
5.3 Limitările load balancingului stateful 34
5.4 Beamer: load-balancing stateless 36
5.4.1 Hashing stabil 37
5.4.2 Daisy chaining 39
5.4.3 Algoritmul din dataplane 40
5.5 Tratarea Multipath TCP 41
5.5.1 Abordarea MPLB 42
5.5.2 Abordarea Beamer 46
5.6 Control plane-ul 47
5.7 Defragmentare 49
5.8 Implementare 51

5.9 Evaluare . . . 52
5.9.1 Micro-benchmarkuri . . . 52
5.9.2 Scalabilitate şi robusteţe . . . 55
5.9.3 Load balancing de HTTP peste MPTCP . . . 59
5.9.4 Scalabilitatea controllerului . . . 61
5.9.5 Hashing stabil . . . 62
5.9.6 Defragmentare . . . 62
5.10 Concluzii . . . 63

6 Prococolul SOCKS, versiunea 6 **65**
6.1 Introducere . . . 65
6.2 Modul de funcţionare . . . 66
6.3 Cereri . . . 68
6.4 Răspunsuri pentru versiuni incompatibile . . . 69
6.5 Răspunsuri pentru autentificare . . . 70
6.6 Răspunsuri pentru operaţiuni . . . 71
6.6.1 Comanda CONNECT . . . 72
6.6.2 Comanda BIND . . . 72
6.6.3 Comanda UDP ASSOCIATE . . . 73
6.7 Opţiuni SOCKS . . . 77
6.7.1 Opţiuni pentru stiva de protocoale . . . 77
6.7.2 Opţiuni care conţin date de autentificare . . . 84
6.7.3 Opţiuni legate de metodele de autentificare . . . 84
6.7.4 Opţiuni legate de sesiuni . . . 86
6.7.5 Opţiuni pentru idempotenţă . . . 88
6.8 Autentificare cu nume de utilizator şi parolă . . . 90
6.9 Autentificare folosind SSL . . . 92
6.10 TCP Fast Open între client si proxy . . . 92
6.11 Starturi false . . . 92
6.12 DNS furnizat de SOCKS . . . 92
6.13 Consideraţii de securitate . . . 93
6.13.1 Cereri mari . . . 93
6.13.2 Atacuri cu replay . . . 94
6.13.3 Epuizarea resurselor . . . 94
6.14 Consideraţii de confidenţialitate . . . 94
6.15 Timing-ul răspunsurilor . . . 94
6.16 Implementare . . . 96
6.17 Evaluare . . . 96
6.18 Concluzii . . . 97

7 Lucrări înrudite **99**

8 Concluzii **101**