Sprawozdanie nr 1 z laboratorium systemów webowych – Podstawowa konfiguracja SQUID, hierarchie serwerów proxy

Jakub Bomba
nr indeksu: 168123
Grupa czw. 13:15, TN
Prowadzący: Anna Kamińska-Chuchmała

Cel ćwiczenia.

Zadaniem było opisanie, jak obsługiwane jest zapytanie przy różnych układach serwerów proxy. W tym celu utworzone zostały wskazane układy. Opisy są wynikiem analizy tego, co pojawiało się w cache oraz zawartości access loga na poszczególnych serwerach pośredniczących.

1. Prosty, dwupoziomowy układ child-parent

W pliku konfiguracyjnym serwerów-dzieci umieszczona została następująca linia:

cache_peer adres_rodzica parent 3128 3130 default

Oczywiście pominięcie parametru default nic by nie zmieniło.

Ważny jest tutaj parametr przedostatni, a raczej jego brak. W przypadku zostawienia tam prezentowanego w pdf-ie dołączonym do zajęć „proxy-only", nasze dzieci tylko i wyłącznie przekierowywały by ruch, nie zapisując nic u siebie. W związku z tym, całą hierarchia nie miała by sensu, gdyż cache byłoby zapisywane tylko u ostatniego rodzica, a w cache dzieci znajdowałyby się tylko nagłówki informujące o trafieniu bądź też nie w rodzica.

Na serwerze squid1, który jest zarówno klientem, jak i pierwszym serwerem-dzieckiem wywołana została komenda pobierająca zawartość statycznej strony deepsloweasy.com:
curl -x localhost:3128 www.deepsloweasy.com

Inną, nawet lepszą opcją, jest odpalenie zapytania z natywnej konsoli systemu. Zamiast localhost podajemy wtedy odpowiednie IP z podsieci host-only. W ten sposób możemy obserwować wszystkie access logi w tym samym czasie przy użyciu funkcji „tail -f". Do tego nie doświadczymy częstego w virtualboxie opóźnienia In/Out.

W efekcie zapytania w access logu pojawił się wpis TCP_SWAPFAIL_MISS oraz informacja o tym, że na serwerze rodzicu również nie ma tej informacji w cache – FIRST_PARENT_MISS/ip_rodzica text/html.

Serwer-rodzic również zachował się w podobny sposób. Tam access log pokazał UDP_MISS oraz TCP_MISS. Po tym fakcie został zapisany TIMEOUT_DEFAULT_PARENT wskazujący, że zasób zostanie pobrany z rodzica, czyli naszego squid3. Squid3 zachował się podobnie jak squid2 – nastąpił UDP oraz TCP miss, jednak potem nastąpiło bezpośrednie pobranie treści strony – DIRECT/ip text/html. Na ostatnim rodzicu w pliku /var/spool/squid/00/00/00000000 zapisana została oryginalna odpowiedź html wraz z nagłówkami. Na pośredniczących serwerach-dzieciach w tym samym katalogu zawarty był plik z nagłówkami informującymi o fakcie pobrania zawartości z hosta-rodzica (pierwsze wywołanie utworzyło takie pliki o typie cache-digest).

Dzieci również oczywiście zapisały u siebie wyjściowy html zwrócony przez żądaną stronę. W tym wypadku warty zaobserwowania jest fakt istnienia nagłówków cache-lookup oraz x-cache mówiących o braku zbuforowanej treści na rodzicach oraz nagłówek Via pokazujący nam całą ścieżkę wywołań. Na squid1 było to odwołanie do squid2, a następnie do jego rodzica – squid3. Był to standardowy przebieg w przypadku, gdy na serwerach-dzieciach nie jest zbuforowana odpowiedź na nasze żądanie.

W przypadku, gdy dziecko posiadało już zapisane odpowiednie cache, access log zwracał tzw. „hita”, została zwracana odpowiedź, a odwoływanie się do rodzica było zbyteczne.

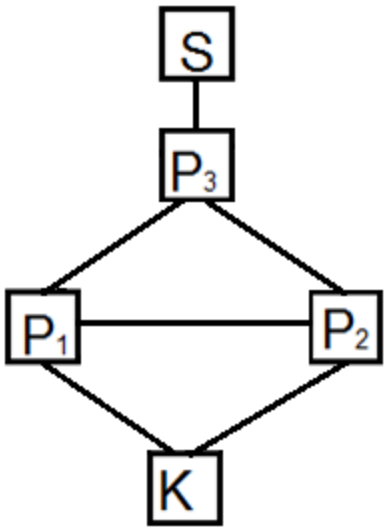
Należy pamiętać, że nie wszystkie strony będą cachowalne. Trzeba zwracać uwagę na ich nagłówki no-store, no-cache.

W dalszej części sprawozdania będą zawarte mniej dokładne informacje, gdyż wszystko działa wg podobnego schematu i nie ma tutaj sensu się powtarzać co do konkretnych zawartości cache, logów oraz nagłówków.

2. Układ z siostrzanymi serwerami-dziećmi posiadającymi tego samego rodzica.

Konfiguracja serwerów dzieci:

cache_peer adres_rodzica parent 3128 3130 default
cache_peer adres_brata sibling 3128 3130 default



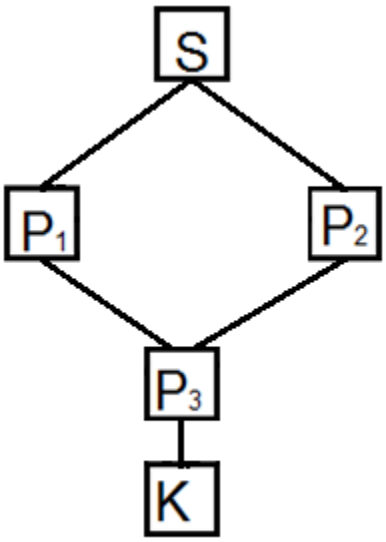
W przypadku tej hierarchii, przy pierwszym uruchomieniu żądania do jednego z dzieci następuje odpytanie rodzeństwa oraz rodzica (ICP QUERY). W przypadku miss-a następuje odwołanie się do rodzica, który w przypadku braku odpowiedzi w buforze stara się ją dokładnie pobrać (jest to poprzedzone UDP oraz TCP missem). Ponowne odwołanie się do rodzica skutkuje zwróceniem odpowiedzi z bufora (UDP_HIT).

W przypadku wprowadzenia opcji proxy-only zarówno dla rodzica, jak i dla rodzeństwa, pobierana zawartość jest od tej jednostki, która zwróci wynik szybciej (zakładamy, że zarówno rodzic jak i rodzeństwo posiadają daną kopię).

Możliwa jest też konfiguracja wymuszająca początkowe odpytanie rodzeństwa oraz (w przypadku braku pozytywnej odpowiedzi) kolejne zwrócenie się do rodzica. Wystarczy wtedy w połączeniu dzieci z rodzicem zawszeć parametr no-query. Nie będzie wtedy od razu wysyłane zapytanie ICP do wszystkich sąsiadów.

3. Jedno dziecko oraz dwójka rodziców.

Konfiguracja dziecka:
cache_peer adres_rodzica1 parent 3128 3130 default
cache_peer adres_rodzica2 parent 3128 3130 default



Jest to standardowa konfiguracja wykorzystująca zapytanie ICP.

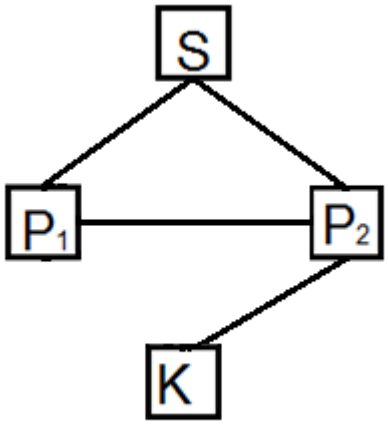
W tym przypadku odpowiedź zwracana jest przez rodzica, który jej szybciej udzieli (jest mniej zajęty). Ciekawym faktem jest to, że ICP pamięta, który rodzic ma już daną stronę w pamięci podręcznej i dla tej samej strony odpytuje tego samego rodzica. W przypadku gdy dwójka rodziców ma daną stronę w cache, odpowiedź zwraca ten szybszy (mniej zajęty w danej chwili). Odpowiedzi rozkładały się tutaj losowo. Oba serwery zwracały UDP_HIT, natomiast tylko jeden odpowiadał przez TCP_HIT i zwracał odpowiedź.

Nieco inaczej wygląda to przy zastosowaniu algorytmu round-robin. Wtedy to odpowiedź jest zwracana losowo, niezależnie czy dany serwer miał daną stronę w pamięci podręcznej, czy też nie. Ważne jest tutaj, aby w parametrze konfiguracyjnym wskazującym na rodzica dodać opcję no-query blokującą żądania ICP.

Obie opisane metody są sposobami na równoważenie obciążenia. W tym przypadku ICP spisuje się jednak lepiej niż round robin. Plusem round robina jest bez wątpienia jego łatwa implementacja oraz prostota w zrozumieniu zasady działania.

4. Dwójka rodzeństwa bez rodziców (sieroty?).

Konfiguracja obu serwerów:
cache_peer adres_brata sibling 3128 3130 default
cache_peer adres_siostry sibling 3128 3130 default



Przy połączeniu się z jednym z dwójki rodzeństwa następuje sprawdzenie, czy odpowiedź znajduje się w buforze. Gdy jest, zostaje po prostu zwrócony TCP_HIT. Gdy jednak bufor jest pusty, odpytane jest rodzeństwo. Gdy tam również nie ma odpowiedzi (UPD oraz TCP MISS), pierwsze dziecko, do którego nastąpiło zapytanie, pobiera zasób bezpośrednio z sieci, po czym go zapisuje buforze. Gdy jednak rodzeństwo ma daną stronę w cache, zwracany jest SIBLING_HIT, czyli dane pobierane są z rodzeństwa. Niestety w rozpisanej hierarchii nie jest możliwe, by rodzeństwo posiadało coś w pamięci podręcznej, gdyż jest ono tylko odpytywane, a nie mając nic po prostu zwraca brak danych i nic nie zapisuje. Sytuacja byłaby ciekawsza, gdyby klient mógł bezpośrednio łączyć się z każdym z rodzeństwa.

Ciekawą sytuację można zaobserwować pytając dziecko, które nie ma nic cache, mając do tego rodzeństwo z zapisaną daną. Oczywiście standardowo zwracany jest SIBLING_HIT. Jednakże w przypadku, gdy owe dziecko zasypiemy żądaniami, będzie ono na tyle często zwracać się do rodzeństwa, że w końcu zwrócony zostanie timeout. W tym wypadku zachowa się ono dość mądrze i pobierze bezpośrednio z sieci zasób (w logu pokazuje się wtedy TIMEOUT_DIRECT), zapisując go do tego w swoim cache. W ten sposób przy kolejnych zapytaniach nie będzie konieczne odpytywanie rodzeństwa, gdyż zasób będzie dostępny lokalnie.

5. Podsumowanie.

Różne rodzaje połączeń pomiędzy serwerami proxy pozwalają na osiągnięcie różnorakich rezultatów, które dają w efekcie mniejsze lub większe obciążenie serwera backendowego. Należy zaznaczyć, że takowe hierarchie należy ustawiać z głową. Przykładowo nie należy łączyć ze sobą jako rodzeństwo dużej ilości serwerów. Zapytanie ICP ma to do siebie, że musi czekać na odpowiedź od wszystkich jednostek, które zostały zapytane. Wiąże się to zawsze z pewnymi opóźnieniami. Nie zaleca się tworzenie hierarchii, gdzie na jednym poziomie występuje ponad szóstka dzieci. Kto chciałby się bowiem opiekować siedmioraczkami. Ciekawa się tutaj może przydać lektura najczęstszych konfiguracji serwera squid, tzw. „Hierarchy tutorial" dostępny na stronie domowej projektu.