Projekt wstępny
19.04.2017 r.

Michał Jarzynka, Grzegorz Kuśmirek, Paweł Masłowski, Sebastian Widemajer

System zbierania statystyk z liczby pakietów (i/lub transmitowanych bajtów) dla najbardziej obciążonych przepływów (z użyciem Netfilter).

System obserwuje predefiniowany zbiór maszyn, w którym inicjuje i zatrzymuje pomiar a następnie zbiera dane. Na każdej obserwowanej maszynie instalowany jest agent, z którym komunikuje się serwer zarządzający. Operator współpracuje z serwerem zarządzającym poprzez: 1) skrypty powłoki, 2) przeglądarkę WWW. Ponadto należy zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów.

1. Nazwa własna projektowanego systemu.

PABS - Packets and Bytes Statistics

2. Opis systemu

Powyższy system ma służyć zbieraniu statystyk z ruchu sieciowego na wybranym wcześniej zbiorze maszyn. Na każdej z owych maszyn będzie zainstalowany odpowiedni program - agent, którego zadaniem będzie nasłuchiwanie rozkazów, zbieranie i przesyłanie statystyk ruchu sieciowego do programu nadrzędnego, obsługiwanego bezpośrednio przez operatora. Program nadrzędny będzie znajdował się na serwerze zarządzającym, do którego dostęp będzie możliwy przez odpowiednie skrypty powłoki lub przeglądarkę WWW.

Analiza wymiany komunikatów między serwerem, a klientami będzie możliwa poprzez stworzony moduł do Wireshark.

3. założenia funkcjonalne i niefunkcjonalne

Wymagania funkcjonalne:

- możliwość inicjowania pomiarów dla wszystkich podłączonych maszyn;
- możliwość inicjowania pomiarów dla każdej z wcześniej zdefiniowanej maszyny z osobna;
- możliwość zatrzymania pomiarów dla każdej z maszyn z osobna i wszystkich łącznie;
- zapisanie wyników pomiaru do wskazanego wcześniej pliku;
- obsługa poleceń z poziomu linii komend (terminala);
- zarządzanie systemem poprzez przeglądarkę WWW;

Wymagania niefunkcjonalne:

- system będzie miał możliwość obsługiwać do 10 agentów jednocześnie;
- Klient wysyła statystyki z ruchu sieciowego zaobserwowanego na danej maszynie okresowo, co 500ms;

- ustalony z góry Timeout 10s;

4. Podstawowe przypadki użycia.

Powyższy system będzie działał w następujący sposób:
- operator wybiera maszyny, na których ma być dokonany pomiar ruch sieciowego;
- operator inicjuje pomiar na wskazanych wcześniej maszynach;
- operator zatrzymuje pomiar w wybranym przez siebie momencie;
- operator może zapisać wyniki pomiarów do pliku;

Zauważmy, że agenci zainstalowani na obserwowanych maszynach, muszą być uruchomieni, żeby w ogóle można było zainicjować pomiar. Początkowo znajdują się oni w stanie nasłuchiwania poleceń z serwera. Dopiero po otrzymaniu odpowiedniego żądania od serwera zarządzającego, przechodzi on w stan “aktywny”, tzn. śledzi ruch sieciowy na danej maszynie i okresowo wysyła do serwera informacje zwrotne.

5. Wybrane środowisko sprzętowo-programowe

Systemy operacyjne:
- Ubuntu 16.04 LTS;
- LMDE 2 Cinnamon 64-bit

IDE:
- Code::Blocks 16.01 - 64bit;
- CodeLite 6.1.1

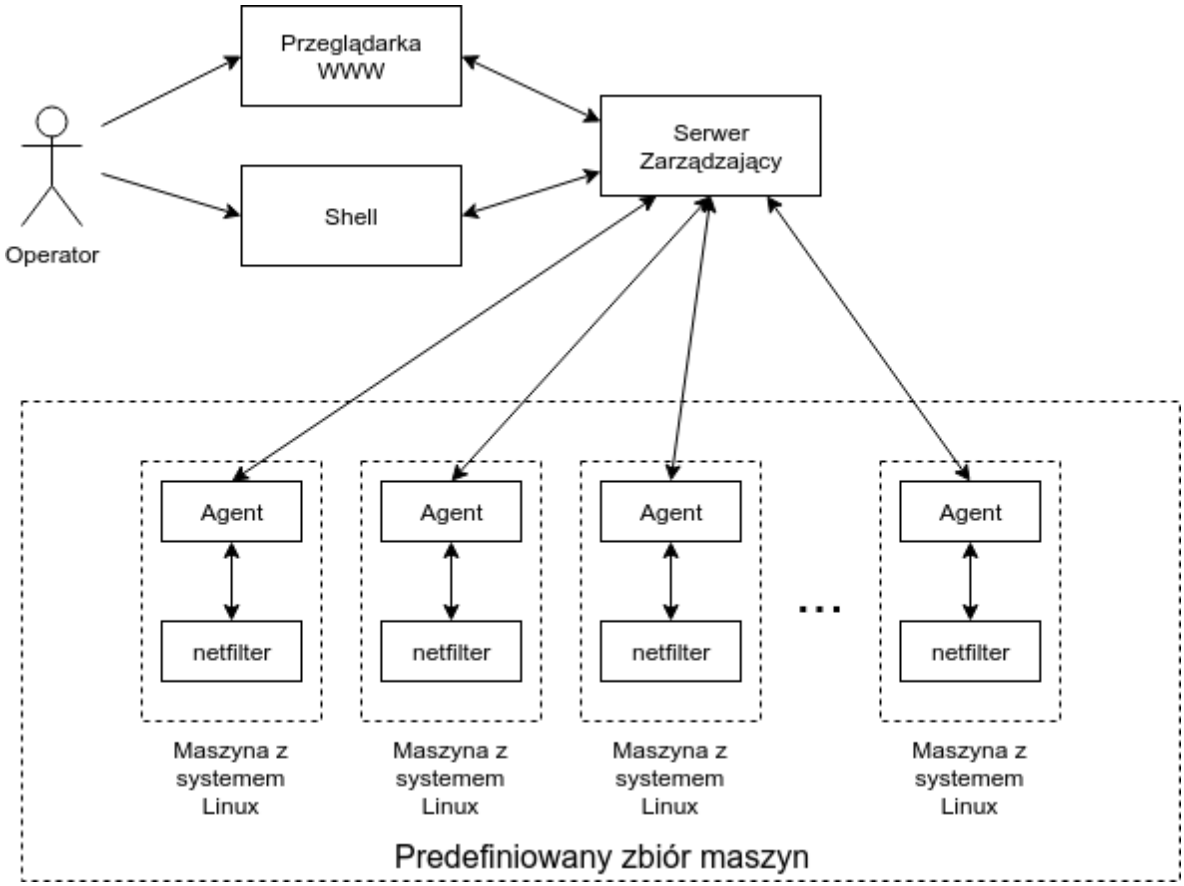
Kompilator:
- GCC 5.3.1
- GCC 4.9.2

Debugger:
- GDB/CDB debugger - debugger domyślny w narzędziu Code::Blocks;
- jw. w narzędziu CodeLite;

Używane biblioteki:

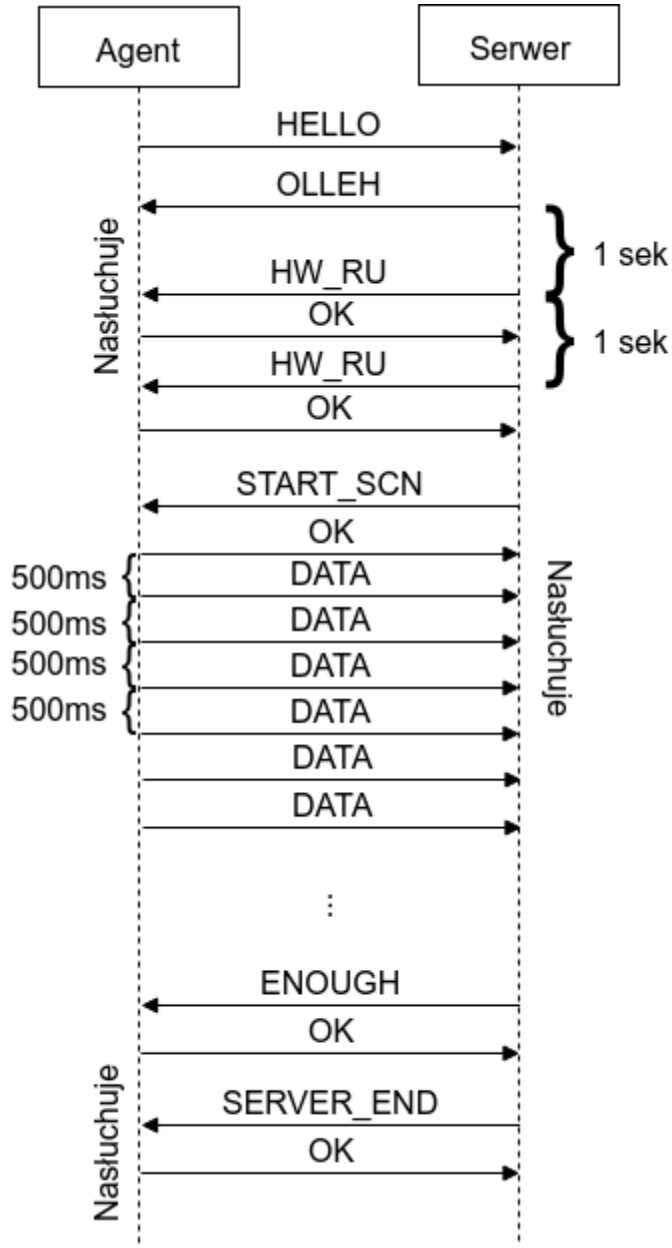
Interesujące nas dane (ilość pakietów oraz bajtów wysyłanych i odbieranych) można otrzymać bez użycia dodatkowych bibliotek z netfilter, używając bezpośrednio wyjścia z iptables przy odpowiednim jego skonfigurowaniu. Alternatywnie, jeśli użycie tych bibliotek jest wymagane to wykorzystane zostaną: libnfnetlink , libnetfilter_queue, libnetfilter_acct.

6. Struktura logiczna systemu:



Operator obsługuje serwer zarządzający za pomocą przeglądarki WWW lub odpowiednich komunikatów przy użyciu powłoki. Serwer zarządzający komunikuje się z agentami zainstalowanymi i uruchomionymi na maszynach z systemem Linux. Agenci w odpowiedzi na żądania serwera mogą wywoływać odpowiednie funkcje netfilter (iptables bezpośrednio lub funkcje biblioteczne), aby zbierać dane o ruchu sieciowym wychodzącym z lub wchodzącym do danej maszyny. Informacje te wysyłane są następnie do serwera, który wyświetla je operatorowi.

Wstępny diagram komunikacji agenta z serwerem:



Opis komunikatów:

- HELLO - nadawane przez uruchomionego agenta, aby rozpocząć komunikację z serwerem
- OLLEH - odpowiedź serwera na komunikat HELLO. Po otrzymaniu tej takiego komunikatu, agent przechodzi w tryb nasłuchiwania.
- HW_RU - Zapytanie serwera sprawdzające czy połączenie jest aktywne, zadawane co 1 sekundę. Po dziesięciu komunikatach bez odpowiedzi następuje timeout i serwer kończy połączenie.
- OK - odpowiedź pozytywna na wszystkie żądania
- ERR - odpowiedź informująca o pojawieniu się błędu. Po takim komunikacie połączenie jest odrzucane.
- START_SCN - żądanie serwera o rozpoczęcie zbierania danych o ruchu przez agenta. Serwer przechodzi w stan odbierania danych.
- DATA - przesyłanie danych o ruchu sieciowym

- ENOUGH - serwer informuje o końcu zbierania danych.
- SERVER_END - serwer kończy działanie i zamyka połączenie
- AGENT_END - agent kończy działanie i zamyka połączenie

7. Sposób testowania

Przy testowaniu wykorzystany zostanie napisany moduł do Wireshark, co pomoże analizować poprawność komunikacji między serwerem a agentem. Ponadto przy testowaniu serwera, posłużymy się programem “atrapą” agenta (napisanym do pierwszego spotkania kontrolnego), który w pełni poprawnie nawiązuje połączenie z serwerem, ale nie ma funkcjonalności zbierania danych o ruchu. Zawsze wysyłać będzie znane wcześniej dane. Podobnie przy implementacji i testowaniu agenta wykorzystany zostanie program “atrapa” serwera.

Aby zaprezentować poprawność działania systemu stworzona zostanie sieć ad-hoc pomiędzy czterema maszynami, z których jedna będzie pełnić rolę serwera. Następnie serwer zainicjuje tryb zbierania danych o ruchu sieciowym, a maszyny z agentami będą pobierać od siebie pliki. Ruch sieciowy zostanie zarejestrowany na serwerze i wyświetlony w przeglądarce lub terminalu. Po pewnym czasie pomiar zostanie zatrzymany a wyniki zapisane do pliku. Powinno to w dostateczny sposób przedstawić poprawność działania wymienionej wcześniej funkcjonalności systemu.

8. Wstępny podział prac

- Sebastian Widemajer - moduł Wireshark i serwer zarządzający
- Paweł Masłowski - obsługa serwera przez przeglądarkę WWW i agent
- Grzegorz Kuśmirek - serwer zarządzający
- Michał Jarzynka - agent

9. Harmonogram prac

10 - 16 maja - moduł Wireshark, komunikacja serwera i agentów (nawiązywanie połączenia, kończenie połączenia, timeout, rozpoczynanie i kończenie zbierania statystyk)

17 - 24 maja - przedstawienie pierwszej wersji skończonego systemu

do 31 maja - przekazanie ostatecznej wersji systemu

10. Adres projektu na serwerze kontroli wersji

https://github.com/TINDreamTeam/PABS