Low Orbit Task Cannon

Techniki Internetowe, Projekt

Tomasz Jakubczyk, Eryk Ratyński, Andrzej Roguski, Kacper Stachyra

23 listopada 2015

Low Orbit Task Cannon na serwerze GitHub

1 Treść zadania

"W sieci jest zbiór zarządzanych węzłów, serwer zarządzający i stacja konsoli administratora. W węzłach pracują agenty zarządzające. Agent zarządzający może: załadować kod nowego procesu, usunąć kod procesu, uruchomić/zatrzymać/wznowić/zabić dany proces zgodnie z harmonogramem, wznowić proces nie raportujący swej żywotności, podać dane statystyczne serwerowi. System umożliwia administratorowi zarządzanie rozproszonymi procesami. System komunikacji powinien móc pracować w przestrzeni adresów IPv4 i IPv6. Ponadto należy zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów."

2 Założenia projektowe

2.1 Środowisko

- Low Orbit Task Cannon (LOTC) uruchamiany jest na systemie operacyjnym GNU/Linux
- LOTC ma stały dostęp do zewnętrznego serwera NTP (w szczególności - łączność z Internetem)

2.2 Zadania

- Zadania po wprowadzeniu do LOTC nie wymaga modyfikacji
- Wykonanie zadania wymaga uruchomienia wyłącznie jednego pliku wykonywalnego (może on jednak uruchamiać inne podprogramy)
- Zadania dają się uruchomić w systemie GNU/Linux bez GUI (w szczególności - bez X Window System)
- Zadania wykonywane są w trybie wsadowym, tj. nie wymagają interakcji z użytkownikiem

3 Struktura systemu

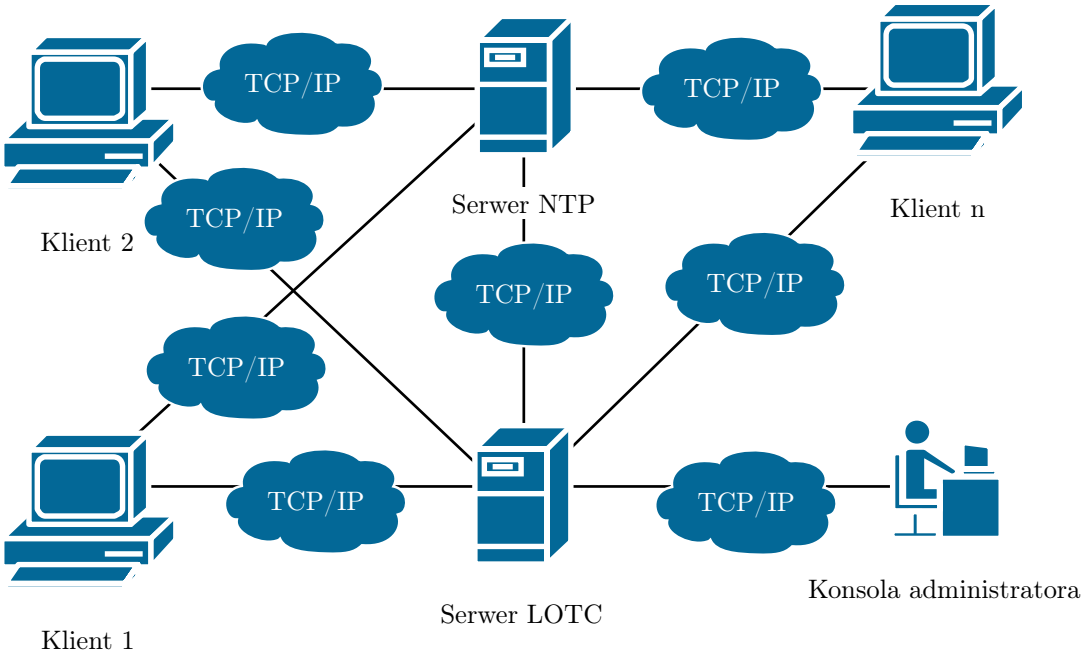
3.1 Moduły

Low Orbit Task Cannon zawiera następujące moduły:

1. Protokół LOTC
2. Serwer
3. Klient (agent)
4. Konsola administratora
5. Minimalny klient NTP
6. Plugin Wireshark (opcjonalny)

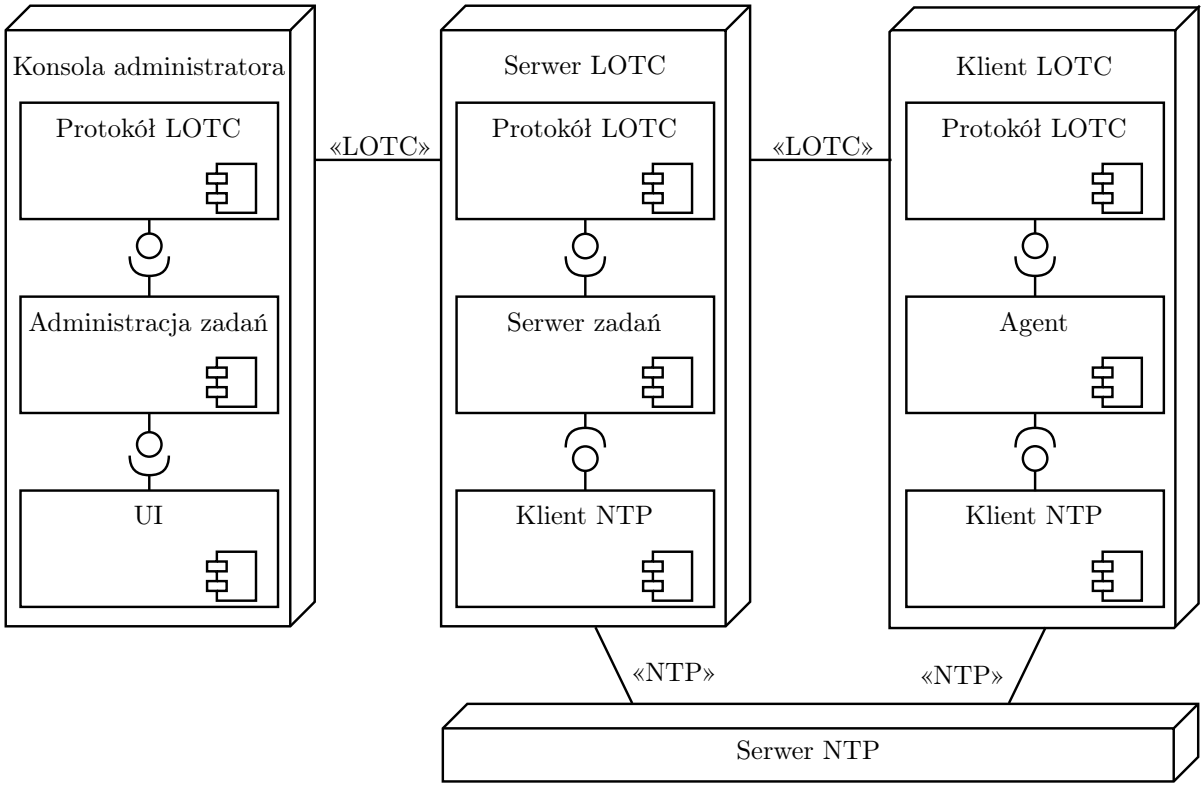
3.2 Topologia

- Każdy klient LOTC musi być połączony siecią TCP/IP z serwerem LOTC i serwerem NTP
- Konsola administratora musi być połączona siecią TCP/IP z serwerem LOTC
- Serwer LOTC musi być połączony siecią TCP/IP z serwerem NTP



3.3 Diagram rozmieszczenia

Komunikacja między konsolą i serwerem oraz serwerem i klientem odbywa się poprzez protokół LOTC. Wykorzystywany jest do tego moduł implementujący protokół LOTC i wystawiający interfejs komunikacyjny.
Z racji potrzeby synchronizacji, serwer oraz klient wykorzystują moduł implementujący niezbędne minimum klienta NTP potrzebne do zapytania serwera NTP o aktualny czas.
Program do administracji zadaniami wystawia interfejs niezbędny do zbudowania UI, zarówno w wersji tekstowej jak i graficznej.



4 Środowisko sprzętowo-programowe

- System operacyjny: GNU/Linux
- Język programowania: C++14 (system LOTC), Lua (Plugin Wireshark)
- Biblioteki: Boost $\geqslant$ 1.59 (z wyłączeniem nakładek na API gniazd BSD)
- Kompilator: GCC $\geqslant$ 5.2.0
- Debugger: GDB $\geqslant$ 7.10

5 Protokół LOTC

5.1 Założenia

Protokół LOTC zapewnia komunikację w warstwie aplikacji między hostami systemu. Podstawowymi zadaniami protokołu są:

1. Rejestracja i usuwanie agentów
2. Zarządzanie wykonaniem pojedynczych zadań
3. Zarządzanie zależnościami między zadaniami
4. Przesyłanie plików
5. Przekazywanie wyników zadań
6. Przekazywanie żądania synchronizacji
7. Monitorowanie responsywności agentów
8. Zgłaszanie błędów

Ponadto protokół powinien weryfikować stan komunikacji poprzez system potwierdzeń i raportów sukcesu/porażki.

5.2 Struktura protokołu

Protokół dzieli się na osiem kategorii, realizujących osiem wyżej wymienionych zadań. Ponadto niektóre z kategorii dzielą się dalej na podkategorie. Ortagonalnie do kategorii funkcjonuje podział komunikatów na:

- Żądania - komunikaty inicjujące wykonanie czynności
- Potwierdzenia - komunikaty potwierdzające otrzymanie żądania
- Raporty sukcesu - komunikaty informujące o pomyślnym wykonaniu czynności
- Raporty porażki - komunikaty informujące o błędzie w trakcie wykonaniu czynności

5.2.1 Kod komunikatu

Każdy komunikat LOTC rozpoczyna się ośmiobitowym kodem jednoznacznie informującym o kategorii, podkategorii i stanie czynności. Kod ma następującą strukturę:

- Kategoria [3 bity]
- Podkategoria [3 bity]
- Stan czynności [2 bity]

5.2.2 Kategorie

Kod binarnie	Kod dziesiątkowo	Kategoria	Opis
000	0	HOST	Rejestracja i usuwanie agentów
001	1	TASK	Zarządzanie wykonaniem pojedynczych zadań
010	2	DEP	Zarządzanie zależnościami między zadaniami
011	3	FILE	Przesyłanie plików
100	4	RET	Przekazywanie wyników zadań
101	5	SYN	Synchronizacja
110	6	PING	Monitorowanie responsywności agentów
111	7	ERR	Zgłaszanie błędów

5.2.3 Podkategorie

Kategoria HOST:

Kod binarnie	Kod dziesiątkowo	Podkategoria	Opis
000	0	H_ADD	Dodanie agenta
001	1	H_RM	Usunięcie agenta
010	2	x	[zabroniony]
011	3	x	[zabroniony]
100	4	x	[zabroniony]
101	5	x	[zabroniony]
110	6	x	[zabroniony]
111	7	x	[zabroniony]

Kategoria TASK:

Kod binarnie	Kod dziesiątkowo	Podkategoria	Opis
000	0	T_ADD	Dodanie zadania
001	1	T_RM	Usunięcie zadania
010	2	T_RUN	Wykonanie zadania
011	3	T_KILL	Zakończenie zadania
100	4	T_STOP	Wstrzymanie zadania
101	5	T_CONT	Kontynuowanie zadania
110	6	x	[zabroniony]
111	7	x	[zabroniony]

Kategoria ERR:

Kod binarnie	Kod dziesiątkowo	Podkategoria	Opis
000	0	E_HEAD	Błędny nagłówek
001	1	E_LGTH	Błędna długość
010	2	E_CSUM	Błędna suma kontrolna
011	3	x	[zabroniony]
100	4	x	[zabroniony]
101	5	x	[zabroniony]
110	6	x	[zabroniony]
111	7	E_OTH	Inny błąd

Pozostałe kategorie:

Kod binarnie	Kod dziesiątkowo	Podkategoria	Opis
000	0	DEF	Wartość domyślna
001	1	x	[zabroniony]
010	2	x	[zabroniony]
011	3	x	[zabroniony]
100	4	x	[zabroniony]
101	5	x	[zabroniony]
110	6	x	[zabroniony]
111	7	x	[zabroniony]

5.2.4 Stany czynności

Kod binarnie	Kod dziesiątkowo	Stan	Opis
000	0	REQ	Inicjacja czynności
001	1	ACK	Potwierdzenie otrzymania komunikatu REQ
010	2	OK	Zakończenie czynności - sukces
011	3	ERR	Zakończenie czynności - porażka

5.2.5 Szczegóły protokołu

Każda z kategorii komunikatów posiada własny nagłówek.

Kategoria HOST:

Pozwala na dodanie albo usunięcie do 2^{16} agentów jednocześnie, obsługuje zarówno IPv4 jak i IPv6.

Pole	Długość [bit]	Opis
Kod komunikatu	8	
Wersja IP	1	Wersja adresów IP agentów 0 - v4, 1 - v6
Wyrównanie	7	Zera
Liczba agentów	16	Liczba agentów
Adres 1	32(IPv4)/128(IPv6)	Adres IP pierwszego agenta (długość pola zależna od wersji IP)
...		
Adres n	32(IPv4)/128(IPv6)	Adres IP n-tego agenta, gdzie n to wartość pola “"”Liczba agentów"

Kategoria TASK:

Pozwala na dodawanie, usuwanie i sterowanie wykonaniem zadań w systemie.

Pole	Długość [bit]	Opis
Kod komunikatu	8	
Flaga priorytetu	1	Określa, czy zatrzymywać zadania o niższym priorytecie
Wyrównanie	7	Zera
Priorytet	16	
ID zadania	32	
Znacznik czasu	32	określa, kiedy polecenie wchodzi w życie

Kategoria DEP:

Pozwala określić, jakie zadania muszą poprzedzać wykonanie zadania Z (maksymalnie 2^{16} zadań).

Pole	Długość [bit]	Opis
Kod komunikatu	8	
Wyrównanie	8	Zera
Liczba zadań poprzedzających	16	
ID zadania Z	32	
ID zadania poprzedzającego 1	32	
...		
ID zadania poprzedzającego n	32	Gdzie n to liczba zadań poprzedzających

Kategoria FILE:

Pozwala przesyłać pojedyncze pliki.

Pole	Długość [bit]	Opis
Kod komunikatu	8	
Flaga typu pliku	1	0 - główny plik wykonywalny zadania, 1 - plik pomocniczy
Długość nazwy pliku	7	Maksymalnie 128 znaków
Suma kontrolna	16	Wyliczana z nazwy i zawartości pliku
ID zadania	32	Określa, do jakiego zadania należy plik
Rozmiar pliku	32	Maksymalnie 2^3 2 bajtów
Nazwa pliku	8 - $8 \cdot 128$	
Zawartość pliku	0 - $8 \cdot 2^{32}$	

Kategoria RET:

Pozwala zwracać wyniki działania zadań.

Pole	Długość [bit]	Opis
Kod komunikatu	8	
Kod zakończenia zadania	8	POSIX exit status
Zwracany typ	8	któryś z typów prostych lub ciąg bitów
Wyrównanie	8	Zera
ID Zadania	32	
Liczba	32/64	Tylko jeśli kod zakończenia = 0 i zwracany typ to liczba
Rozmiar danych	32	Tylko jeśli kod zakończenia = 0 i zwracany typ to ciąg bitów
Dane	0 - $8 \cdot 2^{32}$	Tylko jeśli kod zakończenia = 0 i zwracany typ to ciąg bitów

Kategorie SYN, PING, ERR:

Pozwalają kolejno: zainicjować synchronizację, sprawdzić responsywność i zgłosić błąd w uprzednio otrzymanym komunikacie.

Z racji swojej prostoty, te nagłówki wymagają wyłącznie kodu komunikatu - stan czynności zawarty w kodzie wystarcza do przeprowadzenia niezbędnej komunikacji.

Pole	Długość [bit]
Kod komunikatu	8

6 Serwer

6.1 Opis

Serwer zarządza agentami, które kontrolują procesy na swojej platformie. Informacje o tym gdzie i co ma być zrobione dostaje z konsoli administratora. Kanałem komunikacyjnym z serwerem jest protokół LOTC działający na IPv6 i Ipv4 z wykorzystaniem mechanizmu gniazd sieciowych.

6.2 Reagowanie na zdarzenia

Zdarzeniami przychodzącymi do serwera mogą być polecenia z konsoli administratora, odpowiedzi od agentów lub zdarzenia wewnętrzne serwera np. time-out. Zdarzenia są obiektami z różnych klas typów zdarzeń dziedziczących po bazowej klasie zdarzeń, pojawiającymi się w kolejce zdarzeń. Zostają one obsłużone na podstawie mapy strategii i zdjęte z kolejki. Prawdopodobnie będzie około trzech różnych typów zdarzeń: komunikat z agenta, polecenie od administratora, zdarzenie wewnętrzne.

6.3 Moduły serwera

Serwer da się podzielić na kilka wyraźnych modułów:

- Kontroler – przetwarza zdarzenia przychodzące z modułów i wykonuje lub zleca wykonanie odpowiednich metod w modułach. Odpowiada też za uruchomienie usług modułów szczególnie przy starcie serwera.
- Serwer klienta – odpowiada za komunikację z agentami. Serwer nawiązuje połączenie ze słuchającym agentem, wysyła polecenie, czeka na odpowiedź, otrzymaną odpowiedź wrzuca na kolejkę zdarzeń i zamyka połączenie. Może też nasłuchiwać na połączenia od agenta z wiadomością o zmianie statusu lub okresowym raportem.
- Serwer administratora – odpowiada za komunikację z konsolą administratora. Serwer czeka na polecenie od administratora, wrzuca na kolejkę zdarzeń i zwraca odpowiedź. Konsola administratora może sama nawiązać połączenie i wysłać polecenia, ale jeśli serwer ma po jakimś czasie odesłać raport, to konsola powinna też nasłuchiwać.
- Model – zbiór metod wywoływanych przez kontroler w celu realizacji wybranej strategi z mapy. Może też zawierać maszynę stanów i informacje o agentach.

Poszczególne moduły mogą pracować w osobnych wątkach lub procesach. Kontroler powinien być w wątku głównym.
Model, o ile będzie zgłaszał jakieś zdarzenia (zapewne związane z upływem czasu), powinien mieć osobny wątek. Jeśli nie, to będzie wywoływany (jego metody) w tym samym wątku co kontroler.
Serwer klienta, a właściwie każda jego instancja, powinien być w osobnym wątku. Wątki te powinny być tworzone wtedy gdy wyniknie to ze strategii działania.
Serwer administratora również powinien być w osobnym wątku.

6.4 Komunikacja z konsolą i agentami

Komunikacja z konsolą i agentami odbywać będzie się poprzez protokół LOTC, przy wykorzystaniu modułu dostarczającego interfejs będący nakładką na protokół.

7 Agent (klient LOTC)

7.1 Opis

Agent to autonomiczny program zarządzający przydzielonymi przez serwer zadaniami. Agent zarządzający może: załadować/usunąć zadanie, uruchomić/zatrzymać/wznowić/zabić dany proces zgodnie z harmonogramem, podać dane statystyczne serwerowi.
Agent komunikuje się z serwerem (protokołem działającym na IPv6 i IPv4) wykorzystując mechanizm gniazd sieciowych.

7.2 Zasady działania agenta

Agent po starcie łączy się z serwerem, następnie stale oczekuje na komunikaty wysyłane przez serwer. Po przyjściu komunikatu potwierdza jego otrzymanie a następnie analizuje jego treść według ustalonego protokołu. Potem wywołuje odpowiednią metodę w nowym wątku. Główny wątek dalej czeka na komunikaty od serwera, oraz wysyła raporty z wykonania innych wątków.

7.3 Zarządzanie zadaniami

Agent ma zdefiniowane metody, które będą wykonywane po otrzymaniu odpowiedniego polecenia z serwera. Lista poleceń:

- Załadowanie nowego zadania
- Usunięcie zadania
- Uruchomienie zadania
- Zabicie zadania
- Zatrzymanie wykonywania zadania
- Wznowienie wykonywania zadania
- Synchronizacja czasu
- Potwierdzenie żywotności

8 Konsola administratora

8.1 Opis

Konsola administratora stanowi interfejs, który umożliwia użytkownikowi (administratorowi) sterowanie systemem komunikacji zarządzania rozproszonymi procesami. Poprzez wysyłanie odpowiednich komend, użytkownik powinien mieć możliwość pełnej konfiguracji systemu, wydawania konkretnych poleceń związanych z pracą systemu oraz otrzymywania informacji zwrotnych o statusie całego systemu. Informacje zwrotne można wykorzystać do generowania raportów. Dodatkowo konsola administratora powinna zadbać o poprawność wprowadzanych komend i danych, a także sprawdzać parametry komunikatów otrzymywanych z sieci.

8.2 Komunikacja z serwerem

Komunikacja konsoli administratora z serwerem odbywa się dwustronnie, tj. do serwera są wysyłane komendy i polecenia, a od niego otrzymywane są raporty i informacje zwrotne, a także komunikaty o błędach. Wymaga to ciągłego nasłuchiwania komunikatów z serwera (informację można dostać w dowolnym momencie) przy jednoczesnym rozpoznawaniu komend wydanych przez użytkownika i, po sprawdzeniu ich poprawności, wysłaniu ich na serwer. Wymaga to powołania do pracy dwóch wątków (nasłuchującego i wysyłającego).

8.3 Sprawdzanie poprawności

W programie konsoli administratora będzie sprawdzana poprawność wpisywanych komend, ewentualnych argumentów programu podawanych z linii poleceń (do czego to będzie potrzebne i czy w ogóle?) oraz parametrów komunikatów odebranych z sieci, w celu uniknięcia błędów związanych z działaniem systemu lub celowego złośliwego działania i ataków na system. Na przykład gdy informacje zwrotne z serwera będą zawierały nieprawidłowy identyfikator węzła/procesu zostaną zignorowane, a próba powtórzona.

8.4 Komendy

Komendy wydawane w konsoli administratora powinny umożliwiać:

- Dodanie/usunięcie zadania
- Nadanie priorytetu zadania (przy dodawaniu)
- Definiowanie kolejności następowania zadań (przy dodawaniu)
- Żądanie otrzymania raportu o pracy systemu (jednorazowo bądź cyklicznie)

8.5 Raporty

Konsola administratora, dzięki otrzymywaniu raportów z serwera, w tym raportów cyklicznych, może generować zbiorczy raport z całej pracy systemu (przez określony czas lub do zakończenia pracy) w oddzielnym pliku. Format tych raportów pozostaje do ustalenia (jakie informacje, kiedy, jak często, w jakiej formie to przedstawić).

9 Klient NTP

W celu przeprowadzenia synchronizacji hostów, serwer i klienci korzystają z modułu implementującego minimalistycznego klienta NTP. Moduł ten poza pobraniem aktualnego czasu oblicza też różnicę między czasem systemowym i zwraca parametr korygujący pozwalający uzyskać synchronizację systemu LOTC bez zmieniania zegarów systemowych poszczególnych maszyn.

10 Plugin Wireshark

Plugin Wireshark nie jest częścią systemu LOTC, lecz ma na celu ułatwienie analizy komunikacji między jego hostami. Podstawowymi zadaniami pluginu są:

- Identyfikacja i dysekcja protokołu LOTC
- Zdekodowanie pól nagłówka
- Czytelna prezentacja nagłówka

11 Podział prac

Osoba odpowiedzialna	**Zadania**
Tomasz Jakubczyk	Teamleader, serwer
Eryk Ratyński	Agent
Andrzej Roguski	Protokół, klient NTP, plugin Wireshark, dokumentacja
Kacper Stachyra	Konsola administratora