# 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

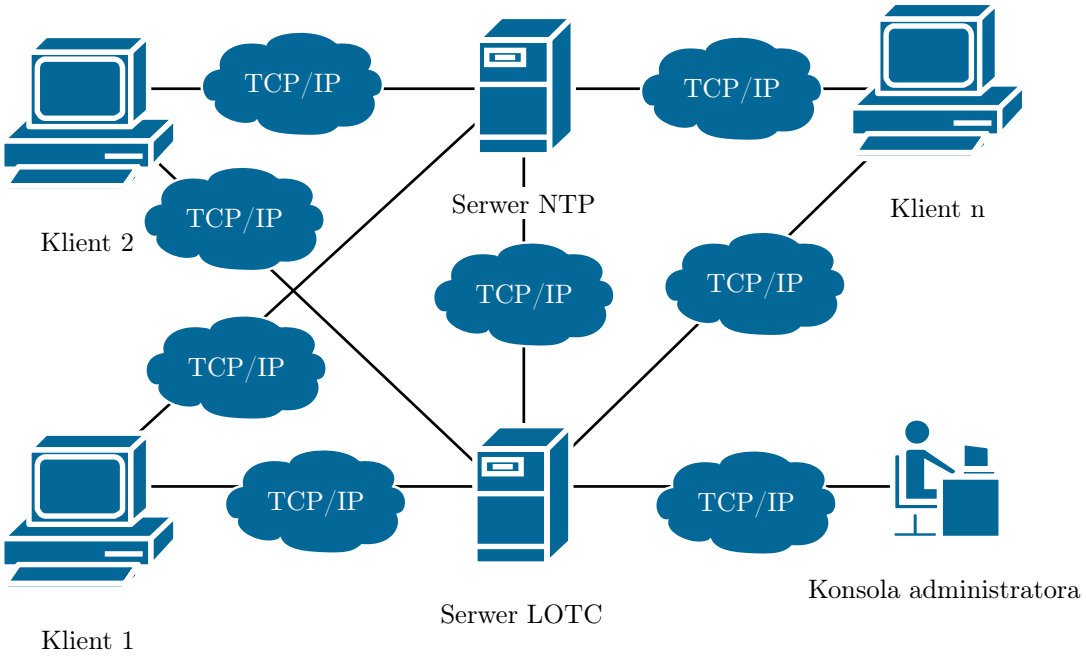
### 3.1 Moduły

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)

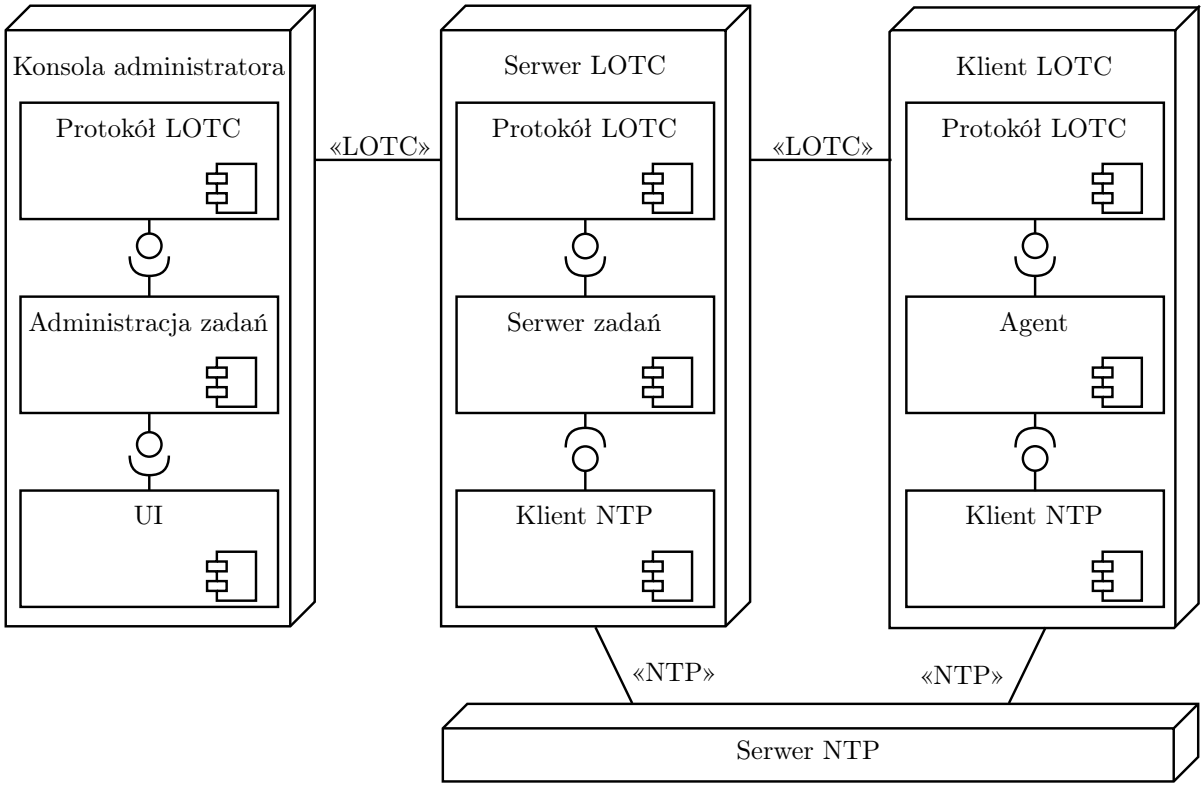
## 3.2 Topologia

- 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

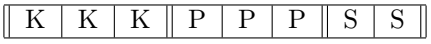
Protokół dzieli się na osiem kategorii, realizujących osiem wyżej wymienionych zadań. Ponadto niektóre z kategorii dzielą się dalej na podkategorie. Ortagonalnie do kategorii funkcjonuje podział komunikatów na:

- 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 | Przekazywanie żądania synchronizacji |
| 110 | 6 | PING | Monitorowanie responsywności agentów |
| 111 | 7 | ERR | Zgłaszanie błędów |

### 5.2.3 Podkategorie

TODO

### 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 |

# 6 Serwer

TODO

# 7 Klient

TODO

## 8 Konsola administratora

TODO

## 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 |