Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

Rok akademicki 2013/2014

Praca dyplomowa inżynierska

Krzysztof Opasiak

Rozproszone monitorowanie systemów komputerowych

Opiekun pracy:
dr inż. Piotr Gawkowski

Ocena .....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia: 1990.12.28

Data rozpoczęcia studiów: 2010.10.01

Życiorys

Urodziłem się 28 grudnia 1990 roku w Koninie. Uczęszczałem kolejno do Szkoły Podstawowej numer 8 im. Powstańców Wielkopolskich w Koninie, a następnie Gimnazjum Towarzystwa Salezjańskiego w Koninie.

W latach 2006-2010 uczęszczałem do Technikum w Zespole Szkół im. Mikołaja Kopernika w Koninie. W trakcie nauki w tej szkole dwukrotnie przyznano mi stypendium Prezesa Rady Ministrów za bardzo dobre wyniki w nauce oraz wzorowe zachowanie. W roku 2010 ukończyłem z wyróżnieniem szkołę średnią, a następnie zdałem maturę oraz egzamin zawodowy, uzyskując tytuł: Technik Teleinformatyk.

W październiku 2010 roku rozpocząłem studia stacjonarne pierwszego stopnia na Wydziale Elektroniki i Technik Informacyjnych na kierunku Informatyka.

.....
podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dn. .....20__r

Z wynikiem .....

Ogólny wynik studiów .....

Dodatkowe wnioski i uwagi Komisji .....

.....

Streszczenie

Przedmiotem pracy jest rozwinięcie systemu rozproszonego monitorowania Icinga pozwalające na monitorowanie urządzeń mobilnych. Przedstawiono przegląd dostępnych na rynku systemów monitorujących oraz wskazano na istotne różnice w charakterze klienta mobilnego i statycznego. Pozwoliło to na zdefiniowanie wymagań stawianych przed projektowanym rozszerzeniem możliwości systemu Icinga. Opisano również jego architekturę i omówiono dostępne do niego dodatki.

W efekcie prac zaproponowano autorskie rozszerzenie funkcjonalności w oparciu o dodatek NSCA oraz wyspecyfikowano własny protokół komunikacyjny. Zaimplementowany w ramach pracy dodatek pozwala na bezpieczne przekazywanie do systemu Icinga danych monitorowanych na urządzeniach mobilnych.

Praca zawiera też opis wykonanej konfiguracji testowej zaprojektowanego systemu. Praktyczność systemu potwierdzono kilkoma istotnymi spostrzeżeniami poczynionymi na podstawie zgromadzonych w systemie monitorującym danych.

Słowa kluczowe: *monitorowanie, systemy rozproszone, systemy mobilne, Icinga, wiarygodność.*

Abstract

Title: *Distributed monitoring of computer systems.*

The subject of this thesis is development of some enhancements of distributed monitoring system Icinga allowing to monitor mobile devices. Review and comparison of some monitoring systems available on the market has been introduced and significant differences between mobile and stationary clients has been pointed out. This allowed to define requirements for some enhancements of Icinga system. Its architecture and some available addons has been also described.

As a result of works, some new Icinga functionality based on NSCA addon with own communication protocol has been introduced. The addon implemented in this thesis allows for secure transfer of monitoring data from mobile devices to Icinga server.

This thesis describes also a test infrastructure created according to the project. Expedience of system has been confirmed with few significant remarks based on analyses of the gathered data.

Key words: *monitoring, distributed systems, mobile systems, Icinga, dependability.*

Serdecznie dziękuję Panu dr inż. Piotrowi Gawkowskiemu za poświęcony czas, a także zaangażowanie i wsparcie okazane podczas pisania tej pracy.

Dziękuję rodzicom oraz dziadkom za wszystko czego mnie nauczyli oraz wsparcie okazane podczas pisania tej pracy, Paulinie Kuśmirek za wyrozumiałość, motywację oraz inspirację. Dziękuję również Justynie Kielniacz, za cenne uwagi oraz wsparcie polonistyczne.

Krzysztof Opasiak

Spis treści

1. Wprowadzenie

Komputery oraz inne urządzenia tworzące infrastrukturę informatyczną przedsiębiorstwa odgrywają bardzo ważną rolę dla procesów biznesowych firmy. Wiele współczesnych ośrodków badawczych, jak i firm, nie może w ogóle funkcjonować, jeśli zostaną pozbawione swojej infrastruktury IT. Znaczna część urządzeń połączona jest ze sobą tworząc sieć prywatną — intranet. Do ich połączenia konieczna jest zarówno rozbudowana struktura okablowania, jak i zestaw urządzeń sieciowych. Brak możliwości używania komputera lub komunikacji poprzez infrastrukturę sieciową niesie ze sobą poważne straty finansowe. Konieczne jest zatem zapewnienie prawidłowego funkcjonowania całej infrastruktury, zawsze, gdy jest ona potrzebna. W zapewnieniu dostępności usług IT istotne jest ciągłe monitorowanie ich stanu oraz przewidywanie i planowanie prac naprawczych i konserwacyjnych. Dzięki temu można uniknąć długotrwałych przerw lub optymalizować ich koszt planując je w godzinach najmniejszego zapotrzebowania na usługi. Istotne jest również wykrywanie z wyprzedzeniem problemów, gdyż pozwala to podejmować odpowiednie akcje zaradcze.

Urządzenia elektroniczne posiadają ograniczoną trwałość, dlatego też istnieje prawdopodobieństwo ich awarii. Ponadto należy pamiętać, iż na urządzeniach uruchamiane jest oprogramowanie, które może zawierać błędy. Z punktu widzenia użytkownika końcowego, za awarię w danym systemie należy zatem uznać nie tylko fizyczne uszkodzenie urządzenia, lecz także sytuację, w której użytkownik zostaje pozbawiony dostępu do danej aplikacji czy usługi. Warto również zauważyć, że użytkownik oczekuje od infrastruktury IT dostarczenia usług o odpowiednim poziomie. Zatem należy również uznać za awarię sytuację, gdy usługi świadczone użytkownikowi nie są na satysfakcjonującym poziomie.

Użytkownik końcowy bardzo często nie jest w stanie udzielić precyzyjnej informacji o usterce, która pojawiła się w jego aplikacji. Często ma miejsce sytuacja, w której użytkownik zgłasza, że program, z którego korzysta, nie działa, natomiast faktyczną przyczyną błędu jest awaria bazy danych lub komunikacji sieciowej. Indywidualna diagnoza przy każdej awarii jest czasochłonna, przez co czas do jej usunięcia wydłuża się, powodując straty finansowe. Konieczne jest zatem monitorowanie stanu urządzeń składających się na infrastrukturę IT przedsiębiorstwa.

Stan urządzenia informatycznego składa się z dwóch elementów. Pierwszym z nich jest stan usług świadczonych przez to urządzenie. Przykładami są nie tylko serwery HTTP czy FTP, lecz również prawidłowe trasowanie pakietów przez router czy filtrowanie ruchu przez zaporę (ang. *firewall*). Sprawdzenie stanu usługi danego urządzenia zazwyczaj może się w łatwy sposób odbywać z innego urządzenia bez konieczności ingerencji w przedmiot badań. Drugim elementem składowym stanu urządzenia są jego parametry wewnętrzne. Przykładami takich parametrów mogą być: obciążenie procesora, zużycie pamięci oraz długość kolejek dyskowych. Parametry urządzenia są zatem jego danymi prywatnymi i ich uzyskanie z zewnątrz jest utrudnione. Do ich pozyskiwania stosuje się zatem specjalne oprogramowanie,

które pozwala na udostępnianie parametrów urządzenia na zewnątrz. Popularnym narzędziem używanym w tym celu jest protokół SNMP (ang. *Simple Network Management Protocol*) [15]. Protokół ten pozwala na dostęp do drzewiastej struktury (MIB — ang. *Management Information Base*), która zawiera parametry urządzenia, jak i pola sterujące. Poprzez mechanizm pułapek (ang. *trap*) możliwe jest również zażądanie notyfikacji, gdy jakiś parametr osiągnie pewną wartość lub wystąpi inne zdarzenie systemowe.

Monitorowanie infrastruktury IT oznacza zatem śledzenie stanów wszystkich jej urządzeń składowych. Zadaniem systemu monitorującego jest nie tylko śledzenie stanu urządzeń lecz również przedstawianie go administratorowi w sposób zgodny z jego oczekiwaniami. Można wyróżnić dwa podstawowe rodzaje monitorowania:

monitorowanie aktywne — rodzaj monitorowania, w którym system monitorujący cyklicznie wykonuje sprawdzenie stanu danego urządzenia lub usługi (ang. *polling*),

monitorowanie pasywne — rodzaj monitorowania, w którym status usługi lub urządzenia zgłaszany jest (być może w nieregularnych odstępach) przez program zewnętrzny do systemu monitorującego.

Każdy ze sposobów monitorowania posiada zarówno swoje wady, jak i zalety. Wybór metody monitorowania zależy zatem od charakterystyki monitorowanego parametru lub usługi. Jeśli podlega on nieregularnym i krótkotrwałym zmianom, a każda z nich powinna być odnotowana, stosuje się monitorowanie pasywne. Klasycznym przykładem zastosowania monitorowania pasywnego jest oczekiwanie na pojawienie się pułapki protokołu SNMP. Nigdy nie wiadomo, kiedy ani ile notyfikacji nadejdzie. Natomiast jeśli dana wartość posiada charakterystykę zmieniającą się w sposób ciągły, należy korzystać wtedy z monitorowania aktywnego, które dokonuje próbkowania danej wartości w określonych odstępach czasu.

Sieci bardzo dużych przedsiębiorstw posiadają budowę wielosegmentową. Ze względów bezpieczeństwa bardzo często składa się ona z sieci wirtualnych, czy wręcz fizycznie odizolowanych od siebie podsieci. Ponadto sieci przedsiębiorstwa bardzo często zawierają zapory ogniowe, które filtrują ruch pomiędzy jej segmentami. Ze względu na fragmentację konieczne jest użycie systemu w architekturze rozproszonej.

Najprostszą realizacją rozproszonego systemu monitorowania jest użycie monitorowania pasywnego do monitorowania wszystkich urządzeń i usług, które nie są widoczne z sieci, w której uruchomiony jest system monitorujący. Niestety może to wymagać zmian w konfiguracji wszystkich urządzeń i uruchomienia na nich dodatkowego oprogramowania. Takie zmiany mogą nie być dozwolone na prostych urządzeniach, których kontrola odbywa się poprzez preinstalowany system producenta.

Możliwa jest również konfiguracja wieloinstancyjnego systemu monitorowania. W każdej odizolowanej komórce sieci należy umieścić instancję systemu, która będzie zbierała dane z tej komórki sieci. Monitorowanie danego fragmentu infrastruktury może się odbywać zarówno w sposób aktywny, jak i pasywny. Po wstępnym przetworzeniu takich danych muszą one zostać zsynchronizowane pomiędzy instancjami, a następnie umieszczone w instancji nadrzędnej lub innym zbiorczym miejscu docelowym. Rozwiązanie to posiada liczne zalety i jest bardzo często stosowane. Dodatkowo niektóre z systemów (np. *Icinga*) umożliwiają wymianę danych pomiędzy instancjami bez konieczności istnienia wyróżnionej instancji nadrzędnej.

Niezależnie od przyjętej architektury monitorowania, samo przedstawienie administratorowi danych bieżących jest często niewystarczające. Precyzyjna diagnoza awarii w możliwe krótkim czasie od jej wystąpienia jest bardzo ważna. Jednak istotna jest również możliwość analizy danych historycznych awarii, aby umożliwić wykrycie potencjalnie krytyczniejszej awarii jeszcze przed jej wystąpieniem. Dostępne są na rynku systemy monitorujące, które pozwalają na gromadzenie informacji w bazach danych. Kolejnym krokiem może być wówczas analiza takiej bazy z wykorzystaniem systemu eksperckiego, który wykaże odpowiednie zależności i na tej podstawie umożliwi wykrycie potencjalnej awarii jeszcze przed jej wystąpieniem.

Współczesne korporacje posiadają nie tylko rozbudowaną infrastrukturę sieciową, lecz również bardzo dużą liczbę urządzeń mobilnych, takich jak laptopy, tablety czy inne urządzenia specyficzne dla danej firmy. Wraz z upowszechnieniem się rozwiązań chmurowych i wirtualizacji stacji roboczych popularność zyskuje idea BYOD (ang. Bring Your Own Device). Bardzo często okazuje się, że poprawne działanie tych urządzeń wpływa znacząco na efektywność pracy osób, które ich używają. Monitorowanie takiego urządzenie jest zadaniem nietrywialnym. Należy pamiętać, iż urządzenie mobilne może nie mieć chwilowej możliwości komunikacji z systemem monitorującym. Jeśli przerwy w łączności występują stosunkowo często, to w przypadku braku późniejszej synchronizacji danych można doprowadzić do fałszywych predykcji systemu eksperckiego. Aby tego uniknąć, konieczne jest lokalne buforowanie danych podlegających analizie i ich dostarczenie, kiedy tylko stanie się to możliwe. W przypadku klienta mobilnego niezwykle istotna jest również kwestia bezpieczeństwa. Urządzenia takie często nie pracują wewnątrz sieci firmowej, lecz używają do komunikacji wielu różnych, niezaufanych sieci. Dane zebrane podczas monitorowania klienta mobilnego mogą zawierać tajemnice handlowe firmy (np. adresację wewnętrzną, nazwy zasobów sieciowych). Konieczne jest zatem zapewnienie zarówno poufności, jak i integralności danych podczas synchronizacji.

W chwili pisania tej pracy, nie udało się odnaleźć na rynku systemu, który umożliwiałby monitorowanie klienta mobilnego zarówno wśród systemów bezpłatnych jak i komercyjnych. Ważne jest dostarczenie odpowiedniego systemu, który pozwoli na kompleksowe monitorowanie wszystkich urządzeń występujących w firmie, zarówno mobilnych, jak i statycznych. W związku z powyższym w niniejszej pracy wykonano rozbudowę popularnego systemu monitorowania *Icinga*, aby umożliwić monitorowanie przy jego użyciu obu typów urządzeń. W ramach pracy zaproponowany został protokół komunikacyjny pozwalający na przekazanie danych z urządzenia mobilnego do systemu monitorującego. Ponadto opracowano i zaimplementowano dodatek do systemu *Icinga* pozwalający na dostarczenie danych do systemu monitorującego używając zaproponowanego protokołu. Praca ta jest częścią systemu monitorowania urządzeń mobilnych opracowywanego na Wydziale Elektroniki i Technik Informacyjnych. Istotnym etapem tej pracy było również utworzenie zaproponowanego systemu monitorowania w środowisku testowym. Ważnym elementem składowym tego systemu jest również praca inżynierska Pana Marcina Kubika [22], w której zawarto opis implementacji aplikacji monitorującej przeznaczonej dla platformy Android. Aplikacja ta wykorzystuje opracowany w ramach niniejszej pracy protokół i dodatek do rejestrowania w systemie *Icinga* zbieranych danych o monitorowanym urządzeniu mobilnym. Na podstawie danych pochodzących z kilkutygodniowego okresu testowego użytkowania systemu potwierdzono konieczność monitorowania zarówno urządzeń stacjonarnych,

jak i mobilnych. Podczas testów wykorzystana została też wcześniej wspomniana aplikacja opracowana przez Pana Kubika.

Układ pracy jest następujący. Rozdział 2 zawiera opis oraz porównanie kilku wybranych systemów monitorowania o otwartym kodzie. W rozdziale 3 przedstawiono problematykę monitorowania klienta statycznego oraz mobilnego. Ponadto po wykonaniu analizy, przedstawiono wymagania, jakie są stawiane przed systemem kompleksowego monitorowania przedsiębiorstwa. Rozdział 4 zawiera opis systemu *Icinga*, na bazie którego budowany jest projektowany system. W rozdziale 5 przedstawiono projekt systemu monitorowania oraz opis protokołu komunikacyjnego. Rozdział 6 zawiera opis wykonanej w ramach niniejszej pracy implementacji dodatku pozwalającego na przekazanie danych o stanie urządzenia pochodzących z monitorowanego urządzenia mobilnego do systemu *Icinga*. W rozdziale 7 znajduje się opis konfiguracji testowej wykonanego systemu, a także sprawozdanie z jego użytkowania. Podsumowanie niniejszej pracy zawarto w rozdz. 8, w którym wskazano również potencjalne możliwości rozwoju wykonanego systemu.

2. Darmowe systemy monitorujące

Na rynku dostępnych jest wiele bardzo różnych systemów monitorujących. Narzędzia z tej grupy możemy podzielić na dwie kategorie:

— Systemy dostępnościowe,
— Systemy analityczne.

Systemy monitorujące, w których główny nacisk położony jest na badanie ciągłej dostępności monitorowanych usług, nazywane są systemami dostępnościowymi. Wspierają one administratora w codziennych zadaniach poprzez nieustanne monitorowanie aktualnego stanu sieci. Narzędzia te są wykorzystywane przede wszystkim do szybkiego powiadamiania administratorów oraz lokalizacji awarii.

Systemy analityczne, w kontekście monitorowania infrastruktury sieciowej, to systemy, które są nastawione na zbieranie i analizę danych o usługach i parametrach urządzeń. Tego typu systemy nie są zazwyczaj wykorzystywane do powiadamiania czy lokalizacji awarii. Ich zadaniem jest przede wszystkim gromadzenie danych dotyczących zużycia poszczególnych zasobów, czy też wskaźników jakości poszczególnych usług. Systemy te posiadają zazwyczaj bardzo rozbudowane narzędzia służące do generacji i analizy wykresów na podstawie zebranych wcześniej danych.

Posiadanie dwóch odrębnych systemów jest niewygodne, zwłaszcza przy monitorowaniu rozbudowanej infrastruktury. Producenci oprogramowania monitorującego wychodząc na przeciw użytkownikom udostępniają dodatkowe komponenty, które zmieniają klasyczny system dostępnościowy w hybrydowy. Pozwalają one na kompleksowe zarządzanie infrastrukturą sieciową. Dzięki zastosowaniu takiego systemu administrator uzyskuje jeden uniwersalny interfejs. Możliwy jest w nim zarówno pogląd bieżącego stanu infrastruktury oraz diagnoza awarii, jak i analiza danych historycznych.

Przechowywanie danych zgromadzonych podczas monitorowania może odbywać się na różne sposoby. Podstawową techniką przechowywania danych, w systemach monitorujących na początku były płaskie struktury plików. Okazało się jednak, że rozwiązanie to jest słabo skalowalne i utrudnia rozmieszczanie systemu na wielu urządzeniach. Ponadto sprawne zarządzanie zgromadzonymi danymi spoza systemu monitorującego wymaga dużego wkładu pracy własnej administratora. Rozpowszechniły się zatem techniki przechowywania zebranych danych w oparciu o bazy danych. Wykorzystuje się tu najczęściej bazy relacyjne i cykliczne[1].

Dane przechowywane w relacyjnych bazach danych zorganizowane są w postaci tabel, a powiązania pomiędzy danymi nazywane są relacjami. Taka organizacja bazy danych sprawia, że wraz z upływem czasu jej rozmiar rośnie. Powoduje to zwiększenie zajętości przestrzeni dyskowej, a także wpływa na czas wykonywania

[1] Istnieje kilka implementacji np. RRD — *Round Robin Database* [14] oraz Whisper [16]. Poszczególne implementacje mogą się znacząco różnić w implementacji oraz dostępnych funkcjonalnościach dodatkowych, jednak ogólna zasada działania jest taka sama.

operacji. Dane są przechowywane w bazie do czasu, gdy użytkownik jawnie je usunie. Pozwala to na przeglądanie dowolnie długiego okresu historii bez utraty dokładności, a także na dynamiczne zarządzanie czasem przechowywania danych. Ponadto możliwa jest w każdej chwili zmiana danych z dowolnego okresu. Narzędzia korzystające z tego typu baz muszą posiadać pewną politykę zarządzania zgromadzonymi danymi aby zapobiec nadmiernej zajętości dysku. Istotne jest jednak, że polityka ta jest uzależniona jedynie od aplikacji, a nie od samej bazy danych i może być zmieniana w dowolnym momencie.

Cykliczne bazy danych posiadają natomiast stały, definiowany podczas tworzenia rozmiar[2]. Rozmiar ten określa liczbę porcji danych, jaka może być przechowywana w bazie. Jeśli rozmiar bazy przekroczy rozmiar zadany przy tworzeniu, wykonywana jest konsolidacja danych. Polega ona na wyliczeniu zadanych wartości w odpowiednich przedziałach i zachowaniu ich w pojedynczych rekordach, oraz usunięcie dokładnych danych. W bazach RRD możliwe są trzy typy konsolidacji danych: minimum, średnia oraz maksimum. Rozmiar bazy danych jest definiowany w chwili jej tworzenia i późniejsza jego modyfikacja nie jest już możliwa. Ponadto należy zwrócić szczególną uwagę na fakt, iż dane są usuwane z bazy danych bez wiedzy użytkownika czy aplikacji, przez co taka baza danych nie może zostać użyta do dokładnej analizy danych historycznych. Istotną kwestią jest, że RDD nie pozwala na modyfikowanie danych historycznych, lecz jedynie tych pochodzących z bieżącego okna czasowego. Oznacza to na przykład brak możliwości importowania do takiej bazy danych historycznych. Istnieją jednak implementacje cyklicznych baz danych, jak na przykład Whisper, które nie posiadają tego ograniczenia.

W dalszych punktach przedstawione zostały wybrane systemy monitorowania. Podstawowym kryterium wyboru opisywanych systemów była ich dostępność na licencji umożliwiającej przeglądanie oraz modyfikację w dowolny sposób jego kodu źródłowego. Ponadto zwrócono również uwagę na popularność systemów w środowisku administratorów, a także na dostępność dokumentacji. Na tej podstawie wybrano spośród systemów analitycznych system *Cacti*. Spośród systemów dostępnościowych wybrano natomiast systemy *Nagios* oraz *Icinga*.

2.1. Cacti

Cacti [2] jest systemem monitorującym rozwijanym przez The Cacti Group Inc. i dystrybuowanym na licencji GPL[3]. System bazuje na oprogramowaniu RRDtool [14]. Jest to narzędzie, które pozwala na wykorzystanie cyklicznej bazy danych RRD do składowania pomiarów wartości w zadanym przedziale czasowym. Ponadto dostarcza ono funkcji do generacji wykresów w kilku formatach. Dokładny opis wszystkich możliwości RRDTool można znaleźć w [14]. Dzięki wykorzystaniu wspomnianego narzędzia system ma bardzo prostą budowę i składa się z następujących elementów:

— interfejsu użytkownika,
— dostawcy danych,

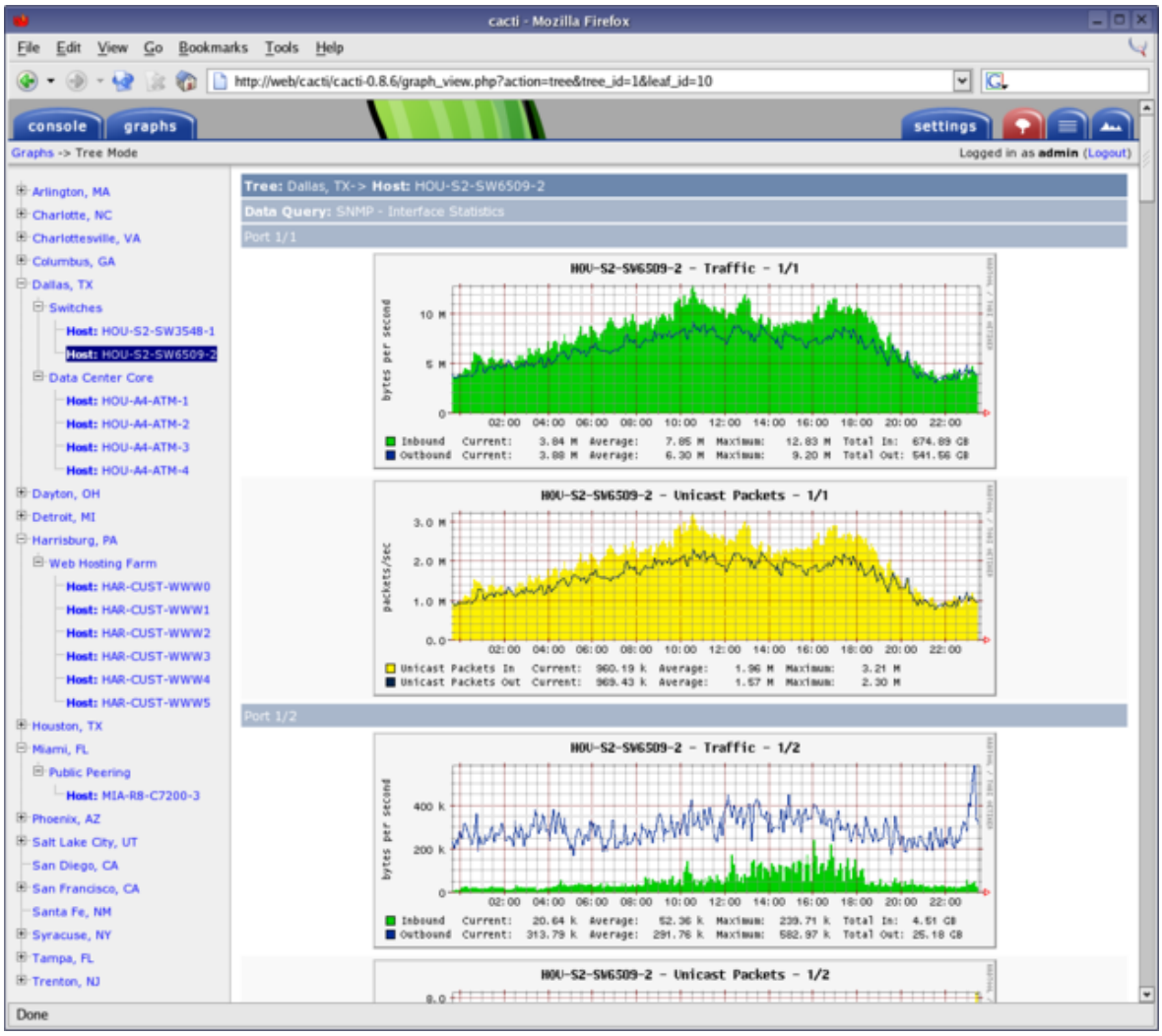
[2] Trwają obecnie prace nad nową implementacją cyklicznej bazy danych — Ceres [9], która znosi to ograniczenie. W chwili pisania tej pracy projekt nie był jednak w fazie rozwoju pozwalającej na jego wykorzystanie.

[3] ang. *General Public License* - popularna licencja oprogramowania o otwartych źródłach. Treść licencji można znaleźć w [18].

— magazynu danych.

Magazyn danych, jest to po prostu cykliczna baza danych RRD obsługiwana przez program przy pomocy narzędzia RRDtool.

Rysunek 2.1. Interfejs użytkownika systemu Cacti.



Interfejs użytkownika został napisany w języku PHP. Do jego działania niezbędny jest serwer http, np. Apache. Rysunek 2.1 przedstawia przykładową podstronę systemu *Cacti*. Z poziomu interfejsu użytkownika możliwa jest graficzna konfiguracja całego systemu. Interfejs posiada klasyczną budowę. Składa się on z jednokolorowego paska menu, w którym zawarte są odnośniki do poszczególnych podstron, oraz z pulpitu, na którym wyświetlane są wybrane dane. Interfejs umożliwia graficzne przedstawienie wyników w postaci wykresów. Format wykresu może być definiowany bezpośrednio przez użytkownika lub można skorzystać z bogatej biblioteki gotowych szablonów. Dostęp do interfejsu zabezpieczony jest przez mechanizm uwierzytelnienia użytkownika systemu monitorującego. Możliwe jest definiowanie wielu użytkowników oraz ich uprawnienia. Każdy użytkownik ma możliwość definiowania własnego zestawu wykresów oraz pulpitów.

Dostawca danych to element systemu, który jest odpowiedzialny za faktyczne wykonywanie sprawdzeń (ang. *check*) aktualnej wielkości danego parametru i przekazywanie ich do narzędzia RRDTool. System umożliwia wybór jednego z dwóch dostawców danych. Pierwszym z nich jest *cmd.php*, który jest prostym skryptem napisanym w języku PHP. Umożliwia on monitorowanie aktywne urządzeń przy pomocy protokołu SNMP. Możliwe jest jedynie wykorzystanie prostego pobierania danych z użyciem tego protokołu. Mechanizm pułapek nie jest obsługiwany. Skrypt *cmd.php* przeznaczony jest do monitorowania jedynie niewielkich sieci. Ze względów wydajnościowych nie jest możliwe wykorzystanie go do monitorowania rozległej infrastruktury.

Drugim z możliwych dostawców danych jest narzędzie *Spine*, nazywane również: *Cactid*. Jest to program napisany w języku C, który uruchomiony jest jako serwis systemowy na urządzeniu monitorującym. Umożliwia on monitorowanie urządzeń zarówno poprzez protokół SNMP, jak i z wykorzystaniem innych metod. Możliwość dostarczenia własnych metod monitorowania opiera się na dostarczeniu skryptu lub pliku wykonywalnego, który będzie cyklicznie uruchamiany przez *Cactid*, a jego wyniki przekazywane w taki sam sposób, jak ze sprawdzeń opierających się na SNMP.

Żaden z dostawców danych nie umożliwia monitorowania danego urządzenia lub usługi w sposób pasywny. *Cacti* nie posiada również żadnego mechanizmu, który pozwoliłby na monitorowanie sieci w sposób rozproszony. Oznacza to, iż administrator musi zmienić konfigurację sieci tak, aby jeden serwer miał dostęp do każdego urządzenia, lub konfigurować i zarządzać osobną instancją w każdym segmencie. Jest to bardzo niewygodne i wręcz uniemożliwia monitorowanie rozległych sieci przy pomocy *Cacti*.

2.2. Nagios

Nagios [8] został opublikowany w 1999 roku na licencji GPL. System od niemal 15 lat jest ciągle rozwijany i udoskonalany zarówno przez autorów, jak i przez szeroką społeczność. W systemie *Nagios* najwyższym priorytetem jest kontrola bieżącej dostępności wszystkich monitorowanych usług. Organizacja systemu zakłada, iż w sieci znajdują się urządzenia (ang. *hosts*), które mogą świadczyć pewne usługi (ang. services)[4]. Każde urządzenie i usługa może znajdować się w jednym z trzech stanów logicznych:

OK usługa działa poprawnie,
WARNING monitorowane parametry przekroczyły stan ostrzegawczy,
CRITICAL parametry usługi przekroczyły stan krytyczny, usługa lub urządzenie nie funkcjonuje.

System posiada rozbudowane algorytmy określania stanu każdego urządzenia oraz usługi. Działanie usługi jest zawsze zależne od stanu urządzenia, na którym dana usługa jest świadczona. Ponadto użytkownik może definiować zależności pomiędzy urządzeniami, np. komunikacja z danym serwerem jest uzależniona od funkcjonowania ruterów znajdujących się na trasie pakietów.

[4] W nomenklaturze systemu *Nagios* usługą (ang. service) nazywana jest zarówno monitorowana usługa zewnętrzna jak np. DNS czy FTP lecz również dowolny parametr tego urządzenia jak zużycie procesora czy pamięci.

Określanie stanu usługi odbywa się przy pomocy zewnętrznych programów nazywanych wtyczkami (ang. *plugin*). W ramach projektu *Nagions Plugins* [7] dostępna jest bardzo duża liczba wtyczek (ponad tysiąc), dzięki czemu system *Nagios* może monitorować w sposób aktywny wszystkie podstawowe usługi. Programy znajdujące się w zestawie pozwalają na badanie usług takich jak HTTP, POP3, SMTP czy FTP oraz pobierać dane o urządzeniu przy pomocy protokołu SNMP. Możliwe jest również napisanie własnych programów lub skryptów, które zostaną wykorzystane jako wtyczki. Konieczne jest jednak, aby programy te spełniały wymagania opisane w [12].

Monitorowanie aktywne usług odbywa się poprzez cykliczne wykonywanie, co zdefiniowany okres czasu, wskazanej wtyczki. Wtyczka otrzymuje dla każdego urządzenia i usługi indywidualny zestaw parametrów wywołania, zdefiniowany przez administratora w plikach konfiguracyjnych systemu. Na podstawie parametrów wywołania oraz aktualnego stanu usługi program określa logiczny stan usługi i przekazuje go do systemu *Nagios* wraz z dodatkowym napisem opisującym stan danej usługi. Napis ten powinien mieć co najmniej jedną linię tekstu i nie więcej niż 4KB. Opis formatowania tego napisu został zawarty w [12].

Możliwe jest również monitorowanie usług w sposób pasywny. Odbywa się ono poprzez dostarczenie przez dowolny program zewnętrzny odpowiednio sformatowanych wyników do systemu monitorującego (wykorzystywany w tym celu jest plik komend zewnętrznych — opisano dalej). Format tych danych jest spójny z formatem przekazywania danych z wtyczki, jednak został on wzbogacony o informacje o czasie wykonania sprawdzenia oraz dane pozwalające na określenie, której usługi i urządzenia dotyczy dane sprawdzenie. Dzięki udostępnieniu tej funkcjonalności system *Nagios* może korzystać na przykład z mechanizmu pułapek w protokole SNMP.

Monitorowanie parametrów urządzeń innych niż to, na którym uruchomiony jest system *Nagios*, wymaga użycia protokołu SNMP lub obecności na danym systemie odpowiedniego agenta. Wraz z systemem *Nagios* dostępnych jest wiele dodatków (ang. addon)[5], które udostępniają takie agenty.

Monitorowanie aktywne tych parametrów może się odbyć na przykład przy pomocy dodatku NRPE (*Nagios Remote Plugin Executor*). Składa się on z dwóch części: agent oraz klient. Agent, jest to demon, który uruchomiony jest na zdalnym (ang. *remote*) urządzeniu jako serwer i oczekuje na żądania od klientów. Klient, jest to natomiast program, który uruchamiany jest przez system *Nagios* jako lokalna wtyczka. W ramach swojego wykonania wtyczka ta łączy się ze wskazanym agentem i zleca mu uruchomienie wskazanej wtyczki na jego maszynie. Istotne jest, że wtyczka, która ma być wykonana musi zostać wcześniej dostarczona przez administratora na urządzenie, na którym znajduje się agent. Wynik wykonania wtyczki na zdalnej maszynie zostanie przesłany do klienta, który przekaże dalej ten wynik do systemu *Nagios*.

Monitorowanie pasywne parametrów urządzenie zdalnego może się odbywać przy pomocy dowolnego programu uruchomionego na tym urządzeniu. Do przekazania wyników tego monitorowania do systemu monitorującego można natomiast

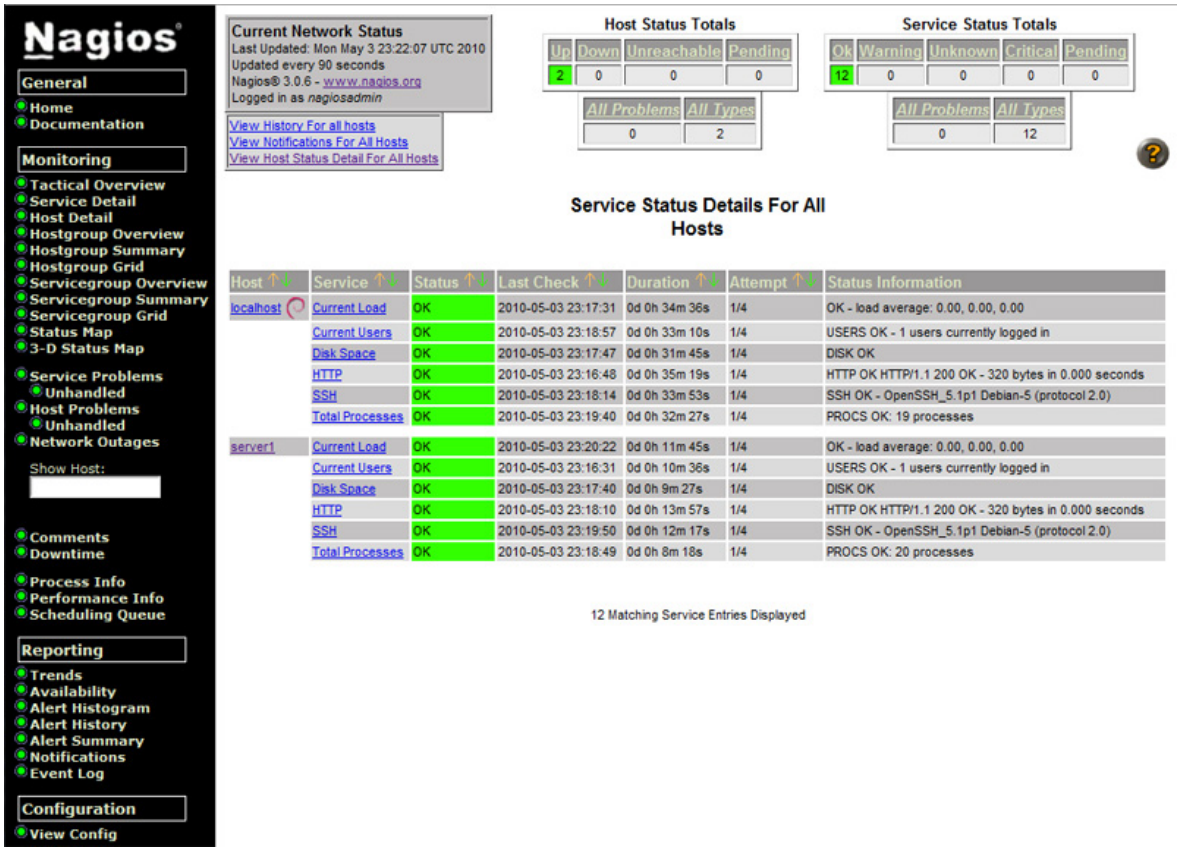
[5] Należy zwrócić uwagę na różne znaczenie słów: "wtyczka"(ang. *plugin*) oraz "dodatek"(ang. *addon*). W nomenklaturze stosowanej w systemie *Nagios* dodatkiem jest każdy program, który zmienia lub rozszerza jego funkcjonalność. Wtyczka natomiast jest to program, lub skrypt pozwalający na sprawdzanie stanu jakiegoś parametru lub usługi.

użyć dodatku NSCA (*Nagios Service Check Acceptor*). Składa się on z dwóch części: serwera oraz klienta. W przeciwieństwie do NRPE, gdzie serwer znajdował się na monitorowanym urządzeniu, w przypadku NSCA serwer uruchomiony jest na tym samym systemie, co *Nagios*. Klient natomiast, uruchamiany jest przez dodatkowy mechanizm monitorujący na zdalnym urządzeniu w celu przekazania wyników pasywnego sprawdzenia do Nagiosa, kiedy jest to potrzebne. Oznacza to, że w porównaniu z dodatkiem NRPE, gdzie agent na urządzeniu mobilnym czekał na polecenia do wykonania, tutaj to serwer NSCA czeka na już gotowe dane ze zdalnych lokalizacji. Szerokie omówienie dodatku NSCA zostało zawarte w rozdz. 4.4.

System *Nagios* posiada rozbudowany system powiadamiania administratora o wystąpieniu awarii oraz o jej zakończeniu lub innych zdefiniowanych wydarzeniach systemowych. Możliwe jest powiadamianie zarówno poprzez email jak i sms. Ponadto możliwe jest automatyczne wykonywanie programów lub skryptów, jeśli wystąpiło jakieś zdarzenie. Podstawowa wersja systemu składa się z następujących elementów:

— interfejs graficzny,
— rdzeń monitorujący,

Rysunek 2.2. Interfejs użytkownika systemu Nagios.



Interfejs graficzny został napisany w języku C z wykorzystaniem technologi CGI[6]. Jego wygląd jest zgodny ze standardami z lat 90: klasyczna strona WWW

[6] *Common Gateway Interface* – znormalizowany interfejs służący do komunikacji pomiędzy serwerem www, a zewnętrznymi programami. Interfejs ten jest wykorzystywany do generowania stron

bez dynamicznie zmieniającej się treści (patrz rys. 2.2). Dane odświeżane są na żądanie klienta lub co określony czas. Wykorzystana technologia zakłada przesyłanie za każdym razem całego dokumentu HTML do klienta, w związku z czym generowany jest nadmierny ruch sieciowy. Widok użytkownika składa się z kliku części. Po lewej stronie widoczne jest klasyczne menu, umożliwiające użytkownikowi wybór treści. Na górze strony natomiast znajduje się podsumowanie aktualnego stanu monitorowanych urządzeń i usług. Centralną część okna zajmuje pulpit, który prezentuje użytkownikowi treść wybraną wcześniej z menu. Interfejs użytkownika umożliwia podgląd aktualnego stanu usług oraz urządzeń. Informacja ta może być wyświetlana w formie listy zawierającej urządzenia i usługi lub w postaci mapy sieci, która pozwala na monitorowanie stanu urządzenia w korelacji z jego logicznym umieszczeniem w strukturze sieciowej. Możliwe jest również przeglądanie historii awarii oraz prostych wykresów stanu urządzenia lub usługi w zadanym przedziale czasu. Dostęp do interfejsu chroniony jest przy pomocy autoryzacji http. Możliwe jest definiowanie wielu użytkowników, jednak tylko z poziomu urządzenia, na którym uruchomiony jest system monitorujący. Należy zauważyć również, że wszyscy użytkownicy danego typu posiadają takie same uprawnienia. Nie ma możliwości dowolnej edycji uprawnień danej grupy czy też użytkownika.

Rdzeń monitorujący został zaimplementowany w języku C. Jest to centrum całego systemu, gdyż zajmuje się on przetwarzaniem wszystkich bieżących danych monitorowania, a następnie składowaniem ich w plikach. Ta cześć systemu jest odpowiedzialna za cykliczne uruchamianie wtyczek, a także przetwarzaniem zarówno wyników ich wykonania oraz wyników monitorowania pasywnego przekazanych do systemu. Dane o stanie sieci przechowywane są w plikach dyskowych. W nomentklaturze systemu *Nagios* noszą one nazwę pamięci podręcznej (ang. *cache*).

Sposób komunikacji pomiędzy rdzeniem monitorującym a interfejsem użytkownika jest bardzo sztywny. Interfejs użytkownika uzyskuje dane o stanie sieci z plików pamięci podręcznej rdzenia monitorującego. Jeśli istnieje konieczność komunikacji w drugą stronę, na przykład ponieważ administrator chce zmienić jakieś parametry, konieczne jest wykorzystanie tzw. pliku komend zewnętrznych (ang. *external commands file*). Jest to potok nazwany, z którego dane pobiera rdzeń monitorujący i na ich podstawie podejmuje odpowiednie działania. Łatwo zauważyć, że metody komunikacji wymuszają funkcjonowanie zarówno interfejsu graficznego jak i rdzenia monitorującego na jednym urządzeniu. Oznacza to również, że nie jest możliwe wykorzystanie jednego interfejsu do wyświetlania danych z kilku równoprawnych instancji, lecz zawsze musi istnieć jedna instancja wyróżniona, która będzie agregowała wszystkie dane.

System posiada rozbudowane możliwości monitorowania rozproszonego. Niestety, do wykonania znacznej części z tych konfiguracji potrzebne są elementy systemu, które są dystrybuowane na licencjach komercyjnych wymagających zakupu praw do korzystania z nich. Sztandarowym przykładem jest Nagios Fusion — komercyjna wersja interfejsu użytkownika. Zapewnia ona możliwość prezentacji w jednym interfejsie informacji pochodzących z wielu instancji systemu. Istnieją również darmowe dodatki czyli programy rozszerzające lub zmieniające funkcjonalność systemu *Nagios*. Przykładem takiego dodatku może być NDOUtils, który pozwala systemowi *Nagios* na wykorzystanie bazy danych MySQL zamiast płaskich

internetowych na żądanie klienta. Zewnętrzny program generuje stronę w języku HTML, a następnie serwer przesyła ją do klienta poprzez serwer http. Szczegółowy opis można znaleźć w [3]

struktur plikowych lub N2RRD, który gromadzi dane z systemu *Nagios* w bazie RRD. Możliwa jest również częściowa integracja systemu *Nagios*[7] z dodatkami lub systemami, które pozwalają na wizualizacje zgromadzonych danych.

2.3. Icinga

System *Icinga* powstał w 2009 roku jako klon (ang. *fork*) systemu *Nagios*. System został wzbogacony o wiele nowych elementów, a także poprawiono wiele błędów obecnych w systemie *Nagios*. Dzięki zachowaniu wstecznej kompatybilności wszystkie wtyczki i dodatki systemu *Nagios* mogą być wykorzystane w systemie *Icinga*. Pozyskano dzięki temu bardzo dużą bazę wtyczek, co umożliwia monitorowanie tych samych usług i urządzeń co protoplasta. Podstawowe zasady funkcjonowania systemu są identyczne, jednak wprowadzono wiele ulepszeń i architektonicznych poprawek.

System *Icinga* został wyposażony w zupełnie nowy interfejs graficzny[8], który zaimplementowano w języku PHP przy użyciu szkieletu aplikacji agavi[9]. Jest on oparty na technologii AJAX, dzięki której komunikacja z użytkownikiem nie opiera się na przesyłaniu całych stron w języku HTML lecz na realizacji żądań generowanych poprzez język skryptowy wykonywany po stronie użytkownika. Dzięki zastosowaniu tej technologi proces wyświetlania strony zużywa mniejszą część pasma, a serwer został odciążony. Nowy interfejs użytkownika jest w pełni dynamiczny i składa się z rozszerzalnego menu po lewej stronie oraz pulpitów użytkownika w centralnej części (patrz rys. 2.3). Możliwe jest otwieranie wielu pulpitów oraz wyświetlanie poszczególnych informacji w osobnych oknach, które można swobodnie przemieszczać w obszarze strony. Przykładowy pulpit dostępny dla użytkownika został przedstawiony na rys. 2.3.

Znacznej zmianie uległ również model bezpieczeństwa. W nowym interfejsie graficznym każdy użytkownik posiada swój zestaw zdefiniowanych uprawnień. Oznacza to, że można ograniczyć użytkownikowi dostęp do danych o konkretnej usłudze lub odmówić wykonywania niektórych czynności administracyjnych. Zarządzanie użytkownikami oraz ich uprawnieniami możliwe jest również z poziomu graficznego interfejsu użytkownika, co znacząco podnosi wygodę użytkowania systemu. Cechy te odróżniają system *Icinga* od systemu *Nagios*, w którym użytkownicy mogli być definiowani jedynie z poziomu systemu operacyjnego, na którym uruchomiony jest system monitorujący, a ich uprawnienia były jednakowe.

Kolejną istotną różnicą jest zmiana sposobu komunikacji pomiędzy rdzeniem monitorującym a interfejsem użytkownika. W systemie *Nagios* komunikacja ta odbywała się poprzez pliki[10], co uniemożliwiało wyodrębnienie interfejsu graficznego i umieszczenie go na oddzielnym urządzeniu. *Icinga* definiuje natomiast dodatkową

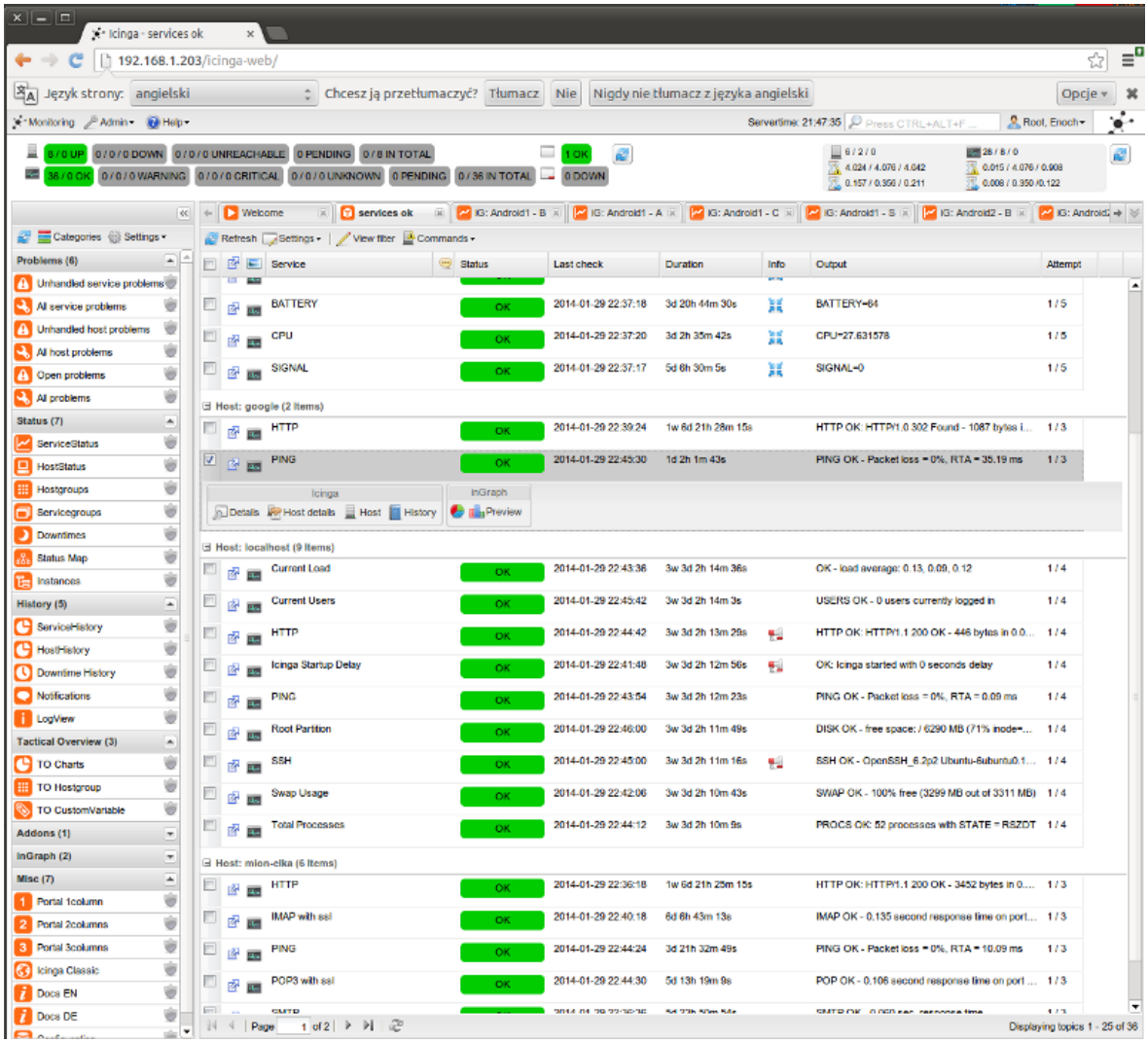
[7] Szczegółowe informacje na temat rozszerzania funkcjonalności oraz samego systemu można znaleźć w [8].

[8] Skorzystanie z nowego interfejsu wymaga użycia modułu IDOUtils oraz bazy danych. Możliwe jest wykorzystanie również klasycznego interfejsu, który nie posiada takich wymagań.

[9] *Agavi* – szkielet aplikacji języka PHP5 pozwalający na łatwą realizację funkcjonalności zgodnej z modelem programowym Model-Widok-Kontroler. Szerszy opis znajduje się na stronie domowej projektu: [1]

[10] Właściwie poprzez potok nazwany i pliki.

Rysunek 2.3. Interfejs użytkownika systemu Icinga.

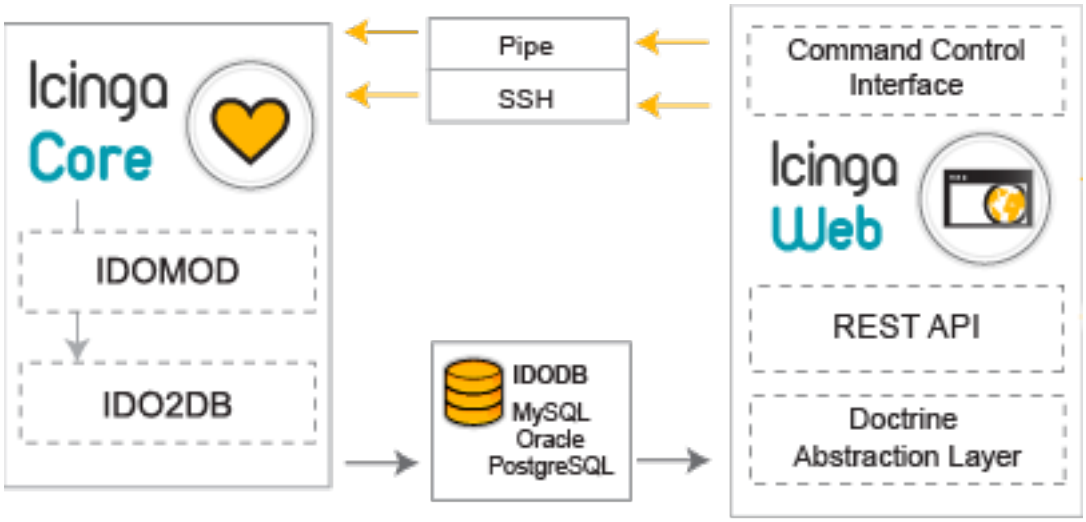


warstwę abstrakcji, która pozwala nowemu interfejsowi użytkownika, na komunikację z rdzeniem monitorującym w sposób niezależny od fizycznej architektury systemu. Schemat komunikacji został przedstawiony na rys. 2.4.

Można zauważyć, że interfejs graficzny w celu pobrania danych o stanie bieżącym systemu nie komunikuje się bezpośrednio z rdzeniem lecz poprzez REST API[11]. Na poziomie implementacji tego API poszczególne wywołania przekładane są na abstrakcyjny język bazodanowy, a następnie, w zależności od typu wykorzystywanej bazy danych, na właściwe pobrania danych z bazy. Wszystkie te operacje są dla interfejsu graficznego czy też innego użytkownika API zupełnie przeźroczyste. Trwają obecnie prace nad modyfikacją API, która umożliwi również przesyłanie żądań do rdzenia monitorującego w ten sam wygodny sposób. Obecny stan pozwala na wysyłanie komend, które mają zostać wykonane poprzez CCI *Command Control Interface*. Implementacja tego interfejsu pozwala na ukrycie faktycznej metody komunikacji z rdzeniem monitorującym.

[11] ang. *Representational state transfer* – lekka metoda przesyłania danych pomiędzy klientem a serwerem.

Rysunek 2.4. Architektura systemu Icinga. Źródło: [10].



Zmiana sposobu komunikacji poszczególnych komponentów systemu, a co za tym idzie całej architektury pozwoliła na uzyskanie modularnej budowy systemu. Taka architektura pozwala na wydzielenie poszczególnych komponentów i umieszczenie ich na odrębnych maszynach. Ponadto warstwa abstrakcji zapewniana przez bazę danych i API pozwala na wyświetlanie przez jeden interfejs informacji zgromadzonych przez wiele instancji rdzenia. Jest to bardzo istotne w przypadku rozległych sieci. Pozwala to na rozdzielenie obciążenia wynikającego z monitorowania na wiele serwerów bez konieczności istnienia instancji nadrzędnej. System *Icinga* posiada również wiele innych konfiguracji rozproszonych, a także redundantnych. Należy również nadmienić, że wszystkie elementy systemu potrzebne do takich konfiguracji są darmowe.

W systemie *Icinga* dopracowano także możliwości współpracy z bazą danych. System ten pozwala na współpracę już nie tylko z bazą MySQL, lecz również z bazami PostgreSQL czy bazą danych firmy Oracle. Możliwość wykorzystania bazy danych Oracle jest bardzo istotna, jeśli dane dotyczące konfiguracji muszą być przechowywane przez bardzo długi czas, lub jeśli monitorowana infrastruktura jest bardzo rozbudowana.

Warto również wspomnieć o dostępnym dla systemu *Icinga* module raportowym opartym na serwerze JasperReports [11]. Pozwala to w bardzo łatwy sposób tworzyć wzory raportów, które następnie będą automatycznie generowane. Umożliwia to regularne i proste tworzenie dokumentacji niezbędnej dla zarządu przedsiębiorstwa. Rozszerzony opis systemu *Icinga* został zawarty w rozdziale 4.

2.4. Podsumowanie

Współczesne systemy monitoringu są bardzo bogato wyposażone i posiadają szereg zaawansowanych możliwości. Każdy z systemów oferuje unikalny zestaw rozwiązań, które z pewnością mogą zostać wykorzystane w wielu instytucjach. Porównując wszystkie omówione systemy, należy zwrócić szczególną uwagę na różnice w ich możliwych zastosowaniach docelowych.

Systemy takie jak *Cacti* zaliczane są do grupy systemów analitycznych. Ich celem jest zatem umożliwienie gromadzenia oraz analizy danych. Zbierane dane mają charakter zagregowany w zadanych przedziałach, na podstawie których prezentowane są użytkownikowi odpowiednie wykresy. Niestety, ze względu na główny sposób gromadzenia danych - protokół SNMP, oraz ubogość innych metod, system ten nie może być wzbogacony o funkcjonalność charakterystyczną dla systemów dostępnościowych.

Drugą grupę systemów stanowią natomiast systemy dostępnościowe, takie jak *Nagios* czy *Icinga*. Ich głównym celem jest monitorowanie bieżącego stanu infrastruktury i raportowanie użytkownikowi najświeższych informacji. Systemy te zostały również zaprojektowane w taki sposób, aby wspomagać administratora w lokalizacji awarii. Głównym typem danych, na których operują te systemy, jest stan urządzenia lub usługi. Zdefiniowanie odpowiednich poziomów kwantyzacji dla stanów pozwala na szybkie uzyskiwanie poglądowych informacji o stanie sieci. Podczas monitorowania gromadzone są również dane szczegółowe. Ich przetwarzaniem nie zajmują się już jednak same systemy monitorowania, lecz liczne dodatki do nich. Możliwe jest zatem rozbudowanie systemu tego typu o dodatkowe elementy, które pozwolą uzyskać system hybrydowy. System taki będzie mógł pełnić rolę zarówno systemu dostepnościowego, jak i analitycznego.

Wybierając system monitorujący należy zatem dokonać szczegółowej analizy wymagań stawianych systemowi. Szczegółowe porównanie wszystkich przedstawionych systemów monitorowania zawarto w tabeli 2.1.

Tablica 2.1: Porównanie systemów monitorowania.

Nazwa systemu	Cacti	Nagios	Icinga
Podgląd stanu bieżącego	Nie	Tak	Tak
Podgląd danych historycznych	Tak	Tak, przez dodatek	Tak, przez dodatek
Dane w formie wykresu	Tak	Tak, przez dodatek	Tak, przez dodatek
Przechowywanie danych w bazie cyklicznej	Tak	Tak, przez dodatek	Tak, przez dodatek
Przechowywanie danych w bazie relacyjnej	Nie	Tak, przez dodatek	Tak, przez dodatek
Powiadomienia o awarii	Nie	Tak, email lub telefon	Tak, email lub telefon
Wsparcie w lokalizacji awarii	Nie	Tak, poprzez mapę logiczną sieci	Tak, poprzez mapę logiczną sieci
Obsługa SNMP	Tak	Tak, przez wtyczkę	Tak, przez wtyczkę

Kontynuacja na następnej stronie.

Tablica 2.1 – Kontynuacja z poprzedniej strony.

Nazwa systemu	Cacti	Nagios	Icinga
Zbieranie danych spoza SNMP	Tak, niewielka liczba dostępnych metod	Tak, bogaty zestaw wtyczek	Tak, bogaty zestaw wtyczek
Monitorowanie pasywne	Nie	Tak	Tak
Nowoczesny interfejs użytkownika	Nie	Nie	Tak, z wykorzystaniem technologi AJAX
Wielu użytkowników	Tak	Tak	Tak
Metoda uwierzytelnienia	Uwierzytelnienie wewnętrzne	Uwierzytelnienie http	Uwierzytelnienie wewnętrzne
Zarządzanie kontami użytkowników z interfejsu	Tak	Nie	Tak
Definiowanie uprawnień dla użytkowników	Tak, przez interfejs graficzny	Nie	Tak, przez interfejs graficzny
Modularność	Nie	Nie	Tak
Rozmieszczenie modułów na różnych urządzeniach fizycznych	Nie dotyczy	Nie dotyczy	Tak
Możliwość monitorowania rozproszonego z instancją nadrzędną	Nie	Tak	Tak
Możliwość monitorowania rozproszonego bez instancji nadrzędnej	Nie	Tak, konieczny płatny dodatek	Tak
Generacja raportów	Nie	Nie	Tak, z wykorzystaniem JasperReports
Możliwość monitorowania urządzenia mobilnego	Nie	Nie	Nie
Dostępność	Darmowy	Częściowo darmowy, wiele płatnych elementów i funkcjonalności	Darmowy

Kontynuacja na następnej stronie.

Tablica 2.1 – Kontynuacja z poprzedniej strony.

Nazwa systemu	Cacti	Nagios	Icinga
Licencja	GPL v2	GPL v3 (tylko darmowe elementy)	GPL v2

Przedstawione systemy monitorujące w znacznym stopniu zaspokajają zapotrzebowanie rynku na systemy monitorowania. Pojawia się jednak pewna nisza związana z monitorowaniem urządzeń mobilnych. Zadanie to nie jest trywialne i wymaga obecności dodatkowych mechanizmów zarówno na urządzeniu mobilnym, jak i w innych elementach systemu. Żaden z analizowanych systemów nie posiadał w swej implementacji, ani w oficjalnych repozytoriach z dodatkami, oprogramowania, które pozwalałoby na monitorowanie parametrów urządzenia mobilnego.

3. Monitorowanie klienta mobilnego

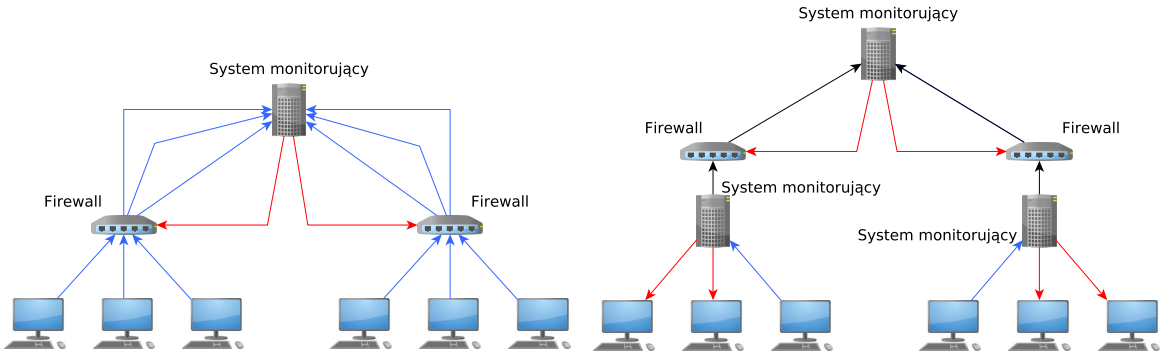
Współczesne systemy posiadają bardzo rozbudowane możliwości monitorowania stanu infrastruktury złożonej z urządzeń sieciowych, stacji roboczych i serwerów. Nowoczesne firmy posiadają jednak coraz większe ilości sprzętu przenośnego, który również należy monitorować. Wzrost liczby urządzeń mobilnych obecnych w infrastrukturach firm wynika nie tylko z zakupów lecz także z popularyzacji podejścia BYOD czyli przynieś swoje własne urządzenie (ang. *Bring Your Own Device*). W firmach, które stosują to podejście pracownicy upoważnieni są do przynoszenia do pracy swoich prywatnych urządzeń, takich jak tablety, telefony czy laptopy. Urządzenia te uzyskują dostęp do wielu chronionych zasobów firmowych przez co konieczne jest zapewnienie w organizacji odpowiedniego poziomu bezpieczeństwa. Wraz z popularyzacją rozwiązań chmurowych pozwalających na realizację idei IaaS (*Infrastructure as a Service*) możliwa jest tania i wydajna wirtualizacja nie tylko serwerów usługowych ale coraz częściej również stacji klienckich czy całych aplikacji. Użytkownik może dzięki temu pracować w tym samym środowisku, niezależnie od urządzenia, które aktualnie wykorzystuje. Zwiększa to elastyczność pracy, jednak niesie również za sobą potrzebę monitorowania urządzeń mobilnych. W przypadku urządzeń firmowych pozwoli to np. na skorelowanie awarii występujących na urządzeniach mobilnych, z tymi, które występują na urządzeniach stacjonarnych, oceny jakości urządzeń różnych producentów itp. W przypadku osób prywatnych monitoring urządzeń mobilnych może służyć wykrywaniu zachowań potencjalnie niebezpiecznych lub kontroli rodzicielskiej. W tym rozdziale przedstawiono charakterystykę monitorowania klasycznej infrastruktury oraz takiej, w której znajdują się urządzenia mobilne. Ponieważ zagadnienie monitorowania tych ostatnich jest rozwijane od niedawna, konieczna jest również definicja wymagań stawianych tego typu systemom.

3.1. Monitorowanie rozproszone klientów statycznych

Poprzez klienta statycznego rozumiemy wszelkie urządzenia stacjonarne wchodzące w skład infrastruktury IT przedsiębiorstwa, które pracują nieprzerwanie lub w dobrze określonych przedziałach czasowych i posiadają dobrze zdefiniowaną hierarchię. Wzajemne relacje pomiędzy tymi urządzeniami wynikają w dużej mierze ze struktury sieci, lecz mogą wynikać także z roli, jaką odgrywają w danej organizacji. Przykładem klienta statycznego mogą zarówno serwery oraz stacje robocze, jak i routery lub urządzenia montażowe przy taśmie produkcyjnej. Dzięki monitorowaniu wszystkich urządzeń w danej sieci systemy monitorujące są w stanie wspierać administratora, wskazując z bardzo dużym prawdopodobieństwem miejsce wystąpienia awarii.

Cechą charakterystyczną systemów monitorowania klienta statycznego jest operowanie zawsze na aktualnych danych. Urządzenie posiadają stałą łączność pomiędzy sobie dlatego wszystkie wyniki sprawdzeń mogą być na bieżąco przetwarzane przez system monitorujący. Każde zerwanie łączności oznacza sytuację awaryjną, która musi być raportowana użytkownikowi.

Rysunek 3.1. Monitoring pasywny (schemat po lewej) oraz rozproszony (schemat po prawej). Kolor czerwony — monitorowanie aktywne, niebieski — monitorowanie pasywne, czarny — komunikacja wewnętrzna systemu.



Sieć w dużej firmie rzadko stanowi jedną całość. Zazwyczaj są to segmenty sieci VLAN oddzielone zaporami lub w ogóle wydzielone fizycznie sieci LAN. Taka separacja urządzeń pozwala na zwiększenie poziomu bezpieczeństwa, lecz jednocześnie utrudnia monitorowanie całej infrastruktury. W celu umożliwienia monitorowania całej sieci firmowej wykorzystuje się monitorowanie rozproszone. Można wyróżnić dwie podstawowe konfiguracje monitorowania rozproszonego (patrz rys. 3.1):

Monitorowanie pasywne — istnieje jedna, centralna instancja rdzenia monitorującego, do którego przesyłane są wyniki sprawdzeń poszczególnych usług. Każde urządzenie samo monitoruje swoje usługi i zgłasza rezultaty.

Wieloinstancyjny system monitorujący — istnieje wiele instancji rdzenia monitorującego. Typowo, każda wydzielona część sieci posiada swoją instancję. Każda instancja może posiadać zarówno usługi monitorowane aktywnie, jak i pasywnie. Wyniki sprawdzeń przesyłane są następnie do jednej wybranej instancji, która gromadzi wszystkie dane lub centralnej bazy danych.

Użycie monitorowania pasywnego dla wszystkich usług jest bardzo niewygodne i jednocześnie utrudnia konfigurację, a także pozbawia administratora możliwości używania niektórych mechanizmów dostępnych wyłącznie dla urządzeń monitorowanych aktywnie. Przykładem takiego mechanizmu może być adaptacyjna częstotliwość wykonywania pomiarów w zależności od zmienności stanu urządzenia. Ponadto wyniki sprawdzeń pasywnych typowo nie są buforowane, lecz wysyłane od razu po ich uzyskaniu. Oznacza to, że jeśli pojawi się chwilowy brak połączenia z serwerem, to dane pomiarów zostaną utracone. W przypadku, gdy jedynym celem systemu jest monitorowanie dostępności danej usługi zewnętrznej serwera, a nie jego parametrów wewnętrznych, jest to jednak błąd pomijalny. Błąd ten staje się jednak istotny, gdy jednym z zadań systemu jest gromadzenie i analiza danych historycznych.

Wieloinstancyjny system monitorujący wymaga zdecydowanie więcej zasobów, jednak pozwala na osiągnięcie znacznie wygodniejszego i bardziej niezawodnego systemu. Ponadto dzięki takiej konfiguracji nie ma potrzeby ingerencji w monitorowane serwery, gdyż możliwe jest monitorowanie ich usług w sposób aktywny. Redukuje to ich obciążenie, a także zwiększa bezpieczeństwo. Warto również wspomnieć, że na przykład system *Icinga* daje możliwość integracji wielu instancji rdzenia monitorującego przy pomocy wspólnej bazy danych. Dzięki temu administrator danej sieci ma możliwość monitorowania i konfigurowania wielu instancji przy pomocy wspólnego interfejsu. Niestety, w systemie Nagios rozwiązanie to zaliczane jest do części korporacyjnej tego systemu, przez co posiada zamknięte źródła i jego wykorzystanie wymaga zakupu licencji. Rozwiązania oparte na istnieniu jednej centralnej instancji rdzenia systemu monitorującego są zazwyczaj darmowe. Wymagają one dodatkowej instancji, zajmującej się agregacją danych co powoduje zwiększenie zużycia zasobów. Należy również zwrócić uwagę, iż niektóre systemy jak Cacti nie posiadają w ogóle możliwości monitorowania pasywnego czy rozproszonego.

3.2. Monitorowanie rozproszone klientów mobilnych

Rosnąca w ostatnich latach popularność technologii mobilnych przyczyniła się do pojawienia się w firmach bardzo dużej liczby urządzeń mobilnych, które również wymagają zarządzania i monitorowania. Urządzenia mobilne są używane bardzo często przez przedstawicieli handlowych oraz menadżerów w celu wykonywania pracy poza obszarem firmy. Duże korporacje coraz częściej decydują się również na wyposażenie swoich pracowników w smartfony lub tablety, które mają ułatwić współpracę z firmą w trakcie podróży służbowych czy spotkań z klientami.

Klient mobilny posiada szereg cech, które znacząco odróżniają go od klientów statycznych. Przede wszystkim należy zauważyć, że urządzenia o których mowa, bardzo często pracują poza obszarem firmy. Wynika z tego, że nie zawsze możliwe jest utrzymywanie takich urządzeń w wirtualnej sieci prywatnej, gdyż urządzenie może znaleźć się w obszarze, gdzie w ogóle nie ma dostępu do Internetu. Ponadto, nie zawsze konieczne jest, aby urządzenia mobilne pracowały podłączone do sieci firmowej. Użytkownicy często wymagają jedynie dostępu do internetu i innych funkcji tego urządzenia. Warto więc zauważyć, że urządzenia te są często narażone na dostęp do sieci o bardzo niskim poziomie zaufania i wielu zagrożeniach. Oznacza to w szczególności, iż urządzenie mobilne zazwyczaj posiada zmienny adres IP, który rzadko jest adresem globalnym. Również struktura sieci, z której korzystają klienty mobilne jest dynamiczna i znajduje się poza obszarem monitorowania administratorów danego przedsiębiorstwa. Oznacza to, że w przeciwieństwie do klienta statycznego, gdzie każda utrata łączności była awarią, utrata łączności jest normalnym elementem działania systemu. Stwarza to konieczność okresowego gromadzenia danych bezpośrednio na urządzeniu mobilnym oraz ich odsyłania w dogodnym czasie.

Znacząca większość klientów mobilnych dzięki kontaktom z siecią pozafirmową posiada, w przeciwieństwie do klientów statycznych, możliwość synchronizacji swojego czasu czy to z serwerami czasu światowego, czy też z sieci GSM. Znaczne rozsynchronizowanie zegarów urządzenia mobilnego z urządzeniami w firmie prowadzić może do istotnych problemów funkcjonalnych (np. brak możliwości uwierzytelnienia, trudność w śledzeniu kolejności zdarzeń). Warto jednak przyjąć pewne

określone zakresy tolerancji czasowych. Wśród serwerów czas jest synchronizowany z dokładnością do milisekund. Taka dokładność w przypadku klienta mobilnego jest zazwyczaj zbędna. Bardzo często istotnymi jednostkami czasu stają się dopiero sekundy lub nawet minuty.

Należy również zwrócić uwagę na duże rozproszenie klientów mobilnych. W przeciwieństwie do klientów statycznych, którzy zazwyczaj pracują w pewnych grupach lub fragmentach sieci, klienty mobilne są zazwyczaj rozpatrywane pojedynczo. Większość klientów mobilnych operuje w pełni samodzielnie, zatem liczność grupy klientów wynosi 1. Istnieją jednak zastosowania, gdzie jeden pracownik użytkuje pewien niewielki zestaw urządzeń przenośnych. Nie zmienia to jednak faktu, że grupa urządzeń mobilnych posiada liczność rzędu maksymalnie kilku urządzeń, a nie nawet do kilkuset jak w przypadku klientów statycznych. W przeciwieństwie do klientów statycznych, gdzie grup koniecznych do wydzielenia było zazwyczaj kilka lub kilkanaście, w przypadku klientów mobilnych takich grup może być kilkaset lub nawet kilka tysięcy.

Warto również dostrzec różnice w sposobie zasilania. Klienty mobilne zazwyczaj posiadają własne zasilanie, przez co każda wykonywana na nim operacja nie tylko spowalnia jego działanie, lecz również zmniejsza jego czas pracy pomiędzy ładowaniami. Przenośność klienta mobilnego zmienia również jego stopień bezpieczeństwa. Urządzenia mobilne stosunkowo często są gubione lub kradzione. W związku z możliwością utraty urządzenia nie powinno się na nim przechowywać tajnych danych, dzięki którym można by skompromitować cały system z którego korzysta klient.

Klient mobilny znacznie różni również zbiorem elementów elementów, które są monitorowane. W przypadku klientów statycznych znaczna część wysiłków jest ukierunkowana na pomiar usług świadczonych przez dany system na rzecz innych systemów lub systemów świadczących określone usługi dla systemu klienckiego (pomiar jakości usługi z punktu widzenia urządzenia klienckiego). Natomiast w przypadku klientów mobilnych istotniejsze wydaje się być monitorowanie parametrów wewnętrznych danego klienta i ewentualnie usług świadczonych przez klienta w ramach grupy klientów mobilnych. Przykładem może być laptop oraz telefon komórkowy, który pozwala mu na dostęp do Internetu. Istotne z punktu widzenia monitorowania są parametry wewnętrzne obu tych urządzeń takie jak stan baterii, siła sygnału itd. oraz jakość usługi — w tym przypadku usługi dostępu do internetu świadczonej przez telefon na rzecz laptopa. Sytuacja ta jest typowa i nie jest zazwyczaj spotykane, aby klient mobilny udostępniał swoje usługi poza grupę klientów, w której on operuje.

3.3. Wymagania dla systemu monitorowania

Klient mobilny posiada zdecydowanie odmienną charakterystykę niż klient statyczny. Dokonano zatem analizy, jakie wymagania należy spełnić, aby dostarczyć system, który sprosta oczekiwaniom administratorów urządzeń mobilnych i statycznych.

Odbiorcą systemu mają być duże firmy i korporacje, które posiadają bardzo rozbudowaną sieć wewnątrz firmy, a ponadto udostępniają swoim pracownikom

urządzenia mobilne różnej klasy. Wśród tych urządzeń znajdują się przede wszystkim telefony oraz tablety z systemem operacyjnym Android lub Windows Phone. Ponadto firma posiada także liczne laptopy wyposażone w system Windows lub Linux. Konieczne jest zatem, aby system pozwalał na monitorowanie każdej ze wspomnianych platform. Duże firmy oraz korporacje zazwyczaj posiadają już oprogramowanie służące do monitorowania swojej infrastruktury sieciowej. Aby umożliwić administratorom łatwe zarządzanie oraz monitorowanie zarówno klientami mobilnymi, jak i statycznymi należy zapewnić integrację systemów monitorowania obu kategorii klientów. Dane odczytywane na urządzeniu mobilnym mogą zawierać dane prywatne pracownika ale i tajemnice handlowe firmy. Oba te rodzaje danych należą do kategorii poufnych i powinny być należycie chronione. Ponieważ urządzenie mobilne będzie pracowało często poza siecią firmową, podczas tworzenia systemu należy zwrócić szczególną uwagę na kwestię bezpieczeństwa przesyłanych danych. Ponieważ system musi przesyłać dane poprzez sieć publiczną, konieczne jest zapewnienie odporności systemu na ataki zewnętrzne oraz na próby przekazywania sfałszowanych danych do systemu. Wszystkie wymagania stawiane omawianemu systemowi zostały zebrane w tabeli 3.1.

Tablica 3.1: Wymagania dla systemu monitorowania klienta mobilnego.

Kod	Nazwa	Opis
W1	Spójność danych	System musi zapewnić, że dane z wykonanych pomiarów nie zostaną utracone, nawet w przypadku ograniczonej łączności. System musi zapewniać spójność danych pomiędzy serwerem, a klientem mobilnym.
W2	Integralności	System musi zapewnić, że wpisy dziennika dostarczone do serwera nie zostały w żaden sposób zmodyfikowane lub dodane.
W3	Autentyczność	System musi zapewnić, że odebrane dane pochodzą od uprawionego klienta.
W4	Poufność	System musi zapewniać poufność danych przesyłanych od klienta poprzez szyfrowanie.
W5	Dodawanie algorytmów	System musi być niezależny od algorytmu kryptograficznego stosowanego podczas przesyłania danych. Ponadto system musi umożliwiać dodawanie w prosty sposób nowych algorytmów kryptograficznych.
W6	Uwierzytelnienie klienta	System musi zapewnić możliwość uwierzytelnienia klienta.
W7	Wymienne algorytmy uwierzytelnienia klienta	System musi być niezależny od algorytmu uwierzytelnienia klienta. Ponadto system musi umożliwiać dodanie w prosty sposób nowych algorytmów uwierzytelnienia klienta.

Kontynuacja na następnej stronie.

Tablica 3.1 – Kontynuacja z poprzedniej strony.

Kod	Nazwa	Opis
W8	Uwierzytelnienie serwera	System musi zapewniać, iż wpisy dziennika zostaną przesłane tylko do wyznaczonego, uprawnionego serwera.
W9	Odporność na zgubienie urządzenia	System musi być odporny na zgubienie urządzenia. Oznacza to, iż zgubienie urządzenia nie może powodować kompromitacji całego systemu.
W10	Dostarczanie w wiele miejsc	System musi umożliwiać przekazywanie danych do wielu podsystemów monitorujących (np. jednocześnie do systemu monitorującego i bazy danych stanowiącej kopię zapasową odbieranych danych), bez konieczności ich retransmisji z klienta mobilnego.
W11	Reguły definiowane dla każdego klienta	System musi umożliwiać definiowanie reguł dotyczących miejsc przeznaczenia dla każdego klienta indywidualnie.
W12	Oszczędność pasma	System powinien minimalizować ilość przesyłanych danych. Ponadto powinien skrócić do minimum czas oczekiwania na potwierdzenie przetworzenia przesłanych danych.
W13	Integracja z istniejącymi systemami	System monitoringu klienta mobilnego musi mieć możliwość integracji i współpracy z istniejącymi systemami monitorowania klienta statycznego.
W14	Analiza danych bieżących	System musi umożliwiać prezentację oraz analizę danych bieżących, a także posiadać możliwość reagowania na wystąpienie zdefiniowanych przez użytkownika zdarzeń.
W15	Analiza danych historycznych	System musi umożliwiać analizę zadanych danych historycznych, włączając w to ich graficzną reprezentację.
W16	Kontrola danych wejściowych	System musi prowadzić kontrolę danych wejściowych od klientów. Konieczne jest, aby system umożliwiał definiowanie, jakie dane mogą być dostarczane przez jakich klientów.
W17	Łatwość dodawania nowych sprawdzeń	System musi umożliwiać dodawanie w łatwy sposób możliwości monitorowania nowych usług i parametrów.
W18	Klient dla platformy Android	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Android.

Kontynuacja na następnej stronie.

Tablica 3.1 – Kontynuacja z poprzedniej strony.

Kod	Nazwa	Opis
W19	Klient dla platformy Windows Phone	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Windows Phone.
W20	Klient dla platformy Windows 8	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Windows 8.
W21	Klient dla platformy Linux	System musi udostępniać klienta pozwalającego na monitorowanie urządzeń opartych na platformie Linux.

3.4. Podstawowe decyzje projektowe

Przedstawione wymagania pozwalają na opracowanie systemu, który zaspokoi potrzebę monitorowania klienta mobilnego. System monitorujący stanowi duży zestaw programów wykonujących się na różnych urządzeniach i w różnych kontekstach. Zaprojektowanie i implementacja od podstaw systemu monitorującego, który spełniłby wszystkie przedstawione wymagania wykracza daleko poza ograniczenia czasowe pracy inżynierskiej. Ponadto dobre praktyki programistyczne nakazują możliwie szerokie wykorzystanie gotowych programów. Należy również pamiętać, iż każdy program wymaga testowania i późniejszego utrzymania jego kodu. Wykorzystanie gotowego systemu pozwala na uzyskanie niskim nakładem czasu systemu, który został już dokładnie przetestowany, a utrzymanie jego kodu zapewniane jest poprzez osoby zewnętrzne. W związku z powyższym w niniejszej pracy podjęto decyzję, aby budowany system monitoringu klienta mobilnego oparty był na jednym z dostępnych darmowych systemów monitorowania.

Na podstawie analizy systemów monitorujących dostępnych na rynku, dokonanej w rozdz. 2, został wybrany system monitorujący *Icinga*. Wybór ten podyktowany jest wieloma zaletami tego systemu. Przede wszystkim należy zauważyć przemyślaną architekturę. System ten posiada budowę modularną, dzięki czemu możliwe jest instalowanie jego komponentów na wielu fizycznych urządzeniach, co umożliwia lepsze zarządzenie obciążeniem serwerów. Umożliwia to wzrost przepustowości całego systemu, a zatem pozwala na monitorowanie bardziej rozbudowanej sieci i infrastruktury w sposób rozproszony. Ponadto posiada on bogaty zestaw wtyczek przeznaczonych do monitorowania wielu popularnych urządzeń i usług. System ten umożliwia zarówno monitorowanie pasywne, jak i wieloinstancyjne. Należy również wyróżnić system *Icinga*, gdyż jako jedyny udostępnia on w sposób darmowy możliwość wspólnego zarządzania i podglądu wieloinstancyjnego systemu monitorującego. Nowoczesny i dynamiczny interfejs użytkownika dostarczany przez ten system może być w łatwy sposób rozszerzany o dodatkowe funkcjonalności. Na szczególne uznanie zasługuje również rozbudowana i na bieżąco aktualizowana dokumentacja projektu. Najważniejszą z zalet jest jednak popularność tego systemu wśród administratorów. Dowodem popularności i wiarygodności systemu *Icinga* może być jego zastosowanie w ośrodku badań Europejskiej Organizacji Badań Jądrowych CERN [19].

Rdzeń monitorujący systemu *Icinga* jest przeznaczony dla systemu Linux, jednak możliwe jest uruchomienie go na większości systemów z rodziny Unix. W związku z powyższym wszelkie rozwiązania zaimplementowane w ramach tej pracy są przeznaczone dla tych samych systemów co rdzeń monitorujący systemu *Icinga*.

Przedstawione wymagania powodują konieczność dokładniejszego zapoznania się z architekturą systemu *Icinga* oraz możliwościami dedykowanych dla niego dodatków. Już na wstępnym etapie projektu można określić, że większość wymagań może zostać spełniona poprzez wykorzystanie odpowiedniej konfiguracji systemu monitorującego. Wysokie wymagania w kwestii bezpieczeństwa oraz zachowania spójności danych powodują jednak, że konieczna jest dokładna analiza systemu oraz dodatków w celu podjęcia decyzji o wykorzystaniu do komunikacji z klientem mobilnym gotowego dodatku do systemu *Icinga* lub zaprojektowaniu i zaimplementowaniu nowego rozwiązania.

Wstępna analiza wymagań wykazała też konieczność implementacji aplikacji mobilnej pozwalającej na monitorowanie danego urządzenia i przesyłanie rezultatów do rdzenia monitorującego. Taka aplikacja przeznaczona dla platformy Android została wykonana przez Pana Marcina Kubika i przedstawiona w [22].

4. System monitorowania Icinga

W rozdziale 2 została przeprowadzona analiza dostępnych na ryku systemów monitorowania. Na jej podstawie dokonano wyboru systemu Icinga jako podstawy do budowy systemu uwzględniającego wymagania dotyczące klienta mobilnego. Przed przystąpieniem do projektowania takiego systemu należy dokładnie zapoznać się z możliwościami jak i ograniczeniami systemu Icinga. W dalszej części tego rozdziału przedstawiono w pełni funkcjonalną konfigurację systemu monitorowania klientów statycznych. Uwzględniono w niej nie tylko system, lecz także dodatki, które będą dla większości konfiguracji niezbędne. W ramach tej konfiguracji uwzględniono również potrzebę analizy zgromadzonych danych. W proponowanym systemie do tego celu użyto dodatku *inGraph*. W odróżnieniu od innych dodatków tego typu, wykorzystuje on relacyjną bazę danych, przez co możliwe jest zarówno dynamiczne zarządzanie czasem przechowywania danych, jak i synchronizacja danych z okresów w przeszłości.

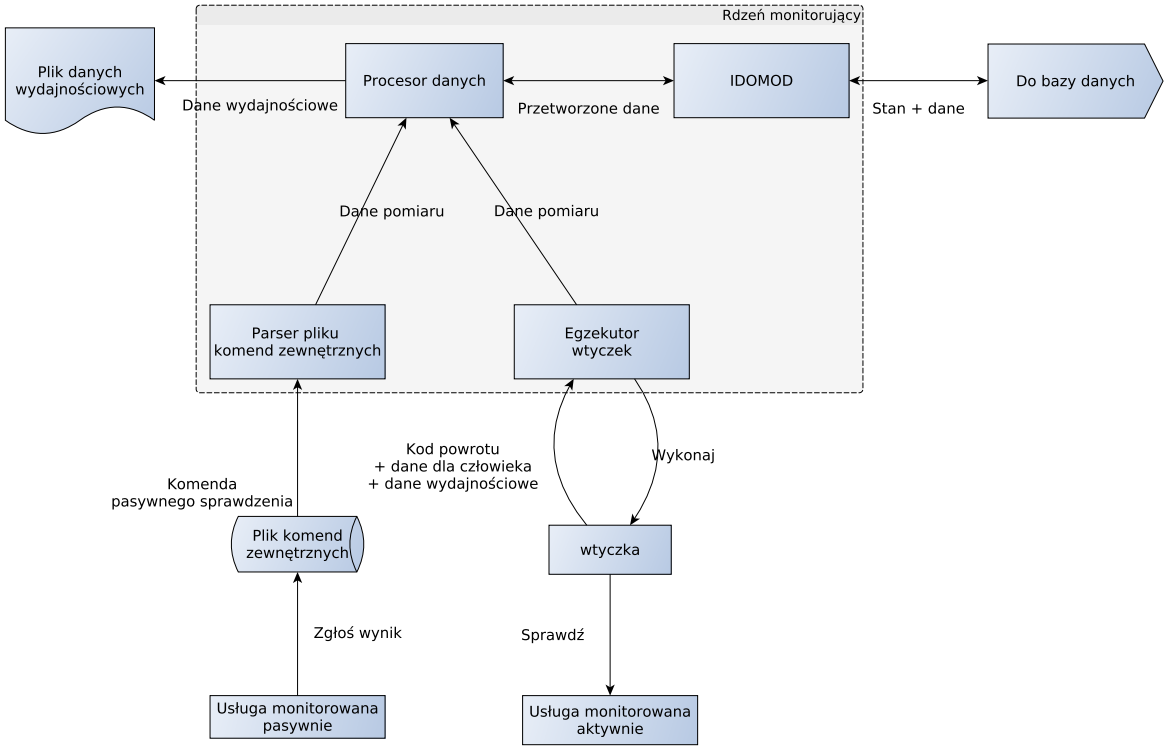
W pierwszej części tego rozdziału opisano poszczególne komponenty, które zostaną wykorzystane w systemie. Druga część tego rozdziału prezentuje proponowaną konfigurację przeznaczoną do monitorowania pojedynczego segmentu sieci. Ponadto przedstawiono możliwości współpracy wielu instancji rdzenia monitorującego podczas monitorowania sieci wielosegmentowej. Konfiguracje te mogą zostać wykorzystane w rozbudowanych sieciach dużych firm i korporacji. Poznanie schematów współpracy wielu rdzeni monitorujących jest kluczowe dla wykonania prawidłowego projektu systemu rozproszonego.

4.1. Rdzeń monitorujący

System *Icinga* powstał jako klon systemu Nagios. Wprowadzono wiele poprawek i zmian architektonicznych jednak zasada funkcjonowania rdzenia pozostała niezmieniona. W celu zrozumienia architektury całego systemu, konieczne jest zapoznanie się z jego głównym elementem czyli rdzeniem monitorującym. Jego architektura została przedstawiona na rys. 4.1. Jest to centralne miejsce w którym wykonuje się przetwarzanie danych o urządzeniach i usługach. Dane, które mają być przetworzone mogą zostać dostarczone na dwa sposoby.

Podstawowy sposób dostarczania danych do przetworzenia opiera się na monitorowaniu aktywnym. W systemie *Icinga* monitorowanie aktywne odbywa się poprzez uruchamianie w określonych odstępach czasowych komend zdefiniowanych przez użytkownika w pliku konfiguracyjnym. Standardowa komenda sprowadza się do uruchomienia wybranego przez administratora programu (tzw. wtyczki) z odpowiednimi parametrami. W ramach wykonania komendy uruchomiony może zostać dowolny program lub skrypt. Aby jednak zapewnić poprawne funkcjonowanie całego systemu, niezbędne jest napisanie programu w zgodności z regułami opisanymi w [12]. Reguły te definiują jakie dane i w jaki sposób powinny być przekazane

Rysunek 4.1. Schemat architektury rdzenia monitorującego systemu Icinga.



z programu (wtyczki) do rdzenia systemu monitorującego. Pierwszym elementem przekazu danych jest zwrócenie przez wtyczkę odpowiedniego kodu zakończenia programu. Poszczególne wartości zwrócone mają następujące znaczenie dla rdzenia monitorującego:

0 — OK, wtyczka mogła wykonać sprawdzenie i usługa lub urządzenie jest w stanie OK.
1 — WARNING, wtyczka mogła wykonać sprawdzenie, ale parametry urządzenia lub usługi przekraczają poziom ostrzegawczy.
2 — CRITICAL, wtyczka mogła wykonać sprawdzenia ale parametry urządzenia lub usługi przekraczają poziom krytyczny.
3 — UNKNOWN, wtyczka nie była w stanie wykonać sprawdzenia ze względu na dostarczenie nieprawidłowych parametrów wywołania lub niskopoziomowego błędu systemu.

Każda wtyczka, poza numerycznym kodem wyjścia z programu, może przekazać do rdzenia monitorującego dane w postaci tekstowej. Minimum powinna zostać przekazana jedna linia tekstu, natomiast maksymalna jego długość wynosi 8KB. Dane te powinny być wypisane na standardowe wyjście wtyczki. Składają się one z dwóch części. Część przed znakiem | (kod ASCII 0x7C) stanowi czytelny dla człowieka opis stanu parametru badanego przez wtyczkę. Część znajdująca się po tym znaku to tak zwane dane wydajnościowe (ang. *performance data*). Powinny być one przekazane w formacie klucz = wartość gdyż są przeznaczona do dalszej obróbki przez zewnętrzne programy np. do generacji wykresów. Przekazanie danych wydajnościowych nie jest obowiązkowe.

Warto nadmienić, iż wszystkie wtyczki uruchamiane przez rdzeń monitorujący wykonują się w kontekście systemu, na którym uruchomiony jest rdzeń monitorujący. W celu wykonywania w sposób aktywny sprawdzeń parametrów innych urządzeń, konieczne jest użycie mechanizmu, który pozwoli na uruchamianie wtyczek na zdalnej maszynie. Przykładem takiego dodatku jest *NRPE* omówiony już w rozdziale 2.2.

Drugą dostępną metodą dostarczenia danych w celu przetworzenia ich przez rdzeń monitorujący jest tzw. plik komend zewnętrznych (patrz rys.4.1). Pozwala on na monitorowanie usług w sposób pasywny. Nie jest to tak na prawdę plik lecz potok nazwany. W rdzeniu monitorującym obecny jest element, który odpowiedzialny jest za czytanie danych z potoku. Do potoku mogą być natomiast zapisywane dowolne spośród komend przedstawionych w [10, 412-436]. Rdzeń monitorujący po przeczytaniu każdej komendy wykona akcję z nią powiązaną. Szczególnym przypadkiem komendy jest żądanie przetworzenia pasywnego sprawdzenia danej usługi lub urządzenia. Dokładny format tej komendy został opisany w [10, 296-299]. Należy zwrócić uwagę na dodatkowe, w stosunku do sprawdzenia aktywnego, pola. Pierwsze z pól to stempel czasu, kiedy zostało wykonane dane sprawdzenie. Kolejne dwie dodatkowe wartości, czyli nazwa urządzenia oraz usługi, konieczne są w celu poprawnej identyfikacji usługi, której dotyczą przekazywane dane. Reszta komendy zawiera dane o znaczeniu znanym już z monitorowania aktywnego.

Wykonanie zapisu do potoku nazwanego należącego do procesu rdzenia monitorującego wymaga, aby program, który chce to zrobić, uruchomiony był na tym samym systemie co rdzeń. Aby umożliwić przekazywanie tych danych z innych urządzeń opracowany został dodatek *NSCA* dystrybuowany jako dodatek do systemów Nagios oraz *Icinga*. Specyfikuje on protokół komunikacyjny, który pozwala na przekazanie z innego systemu do rdzenia monitorującego wiadomości zawierającej wynik sprawdzenia. Program ten został szeroko opisany w rozdz. 4.4.

Otrzymane w ten sposób dane są w kolejnym etapie przetwarzane przez rdzeń sprawdzający niezależnie od sposobu ich dostarczenia (patrz rys. 4.1). Pierwszym etapem przetwarzania tych danych jest wydzielenie z nich danych przeznaczonych dla człowieka oraz danych wydajnościowych przeznaczonych do przetwarzania przez inne dodatki. Dane wydajnościowe zawierają wyniki pomiarów przeprowadzonych przez wtyczkę w trakcie determinowania jej stanu. Po wydzieleniu zostają one udostępnione na zewnątrz rdzenia monitorującego poprzez pliki tekstowe o zadanym w konfiguracji formacie. Dane te są wykorzystywane przez dodatki przeznaczone do generacji wykresów takie jak opisany w rozdz. 4.3 dodatek *inGraph*. Poza eksportem danych wydajnościowych, w ramach przetwarzania dokonywane jest wyznaczenie stanu usługi na podstawie danych poprzednich oraz nowo dostarczonych. Po wykonaniu całego niezbędnego przetwarzania wszystkie dane otrzymane od wtyczki są przekazywane do modułu rdzenia IDOMOD, będącego częścią komponentu *IDOUtils*. Komponent ten pozwala na przechowywanie danych operacyjnych systemu monitorującego w bazie danych. Został on szczegółowo opisany w rozdz. 4.2.

4.2. Komponent IDOUtils

Komponent *IDOUtils* czyli *Icinga Database Object Utilities* jest to zestaw programów, dzięki którym możliwe jest składowanie w bazie danych informacji generowanych przez rdzeń monitorujący. W wersji dostępnej podczas pisania tej pracy wspierane były następujące systemy zarządzania bazą danych:

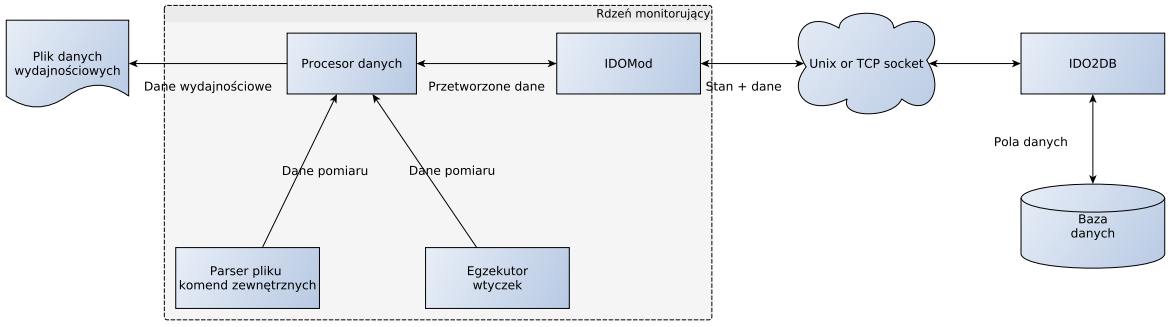
— MySql,
— PostgreSQL,
— Oracle.

W celu zapewnienia funkcjonalności omawianego komponentu, konieczne jest utworzenie bazy danych o odpowiednim schemacie, który został opisany w [10, 669-750]. Udostępnione zostały również skrypty SQL, które definiują wymagane tabele. Ponadto administrator musi zapewnić odpowiednią konfigurację bazy danych, w tym konto użytkownika i hasło w taki sposób, aby umożliwić odpowiednim elementom komponentu *IDOUtils* dostęp do bazy danych.

W celu odciążenia komputera, na którym uruchomiony jest rdzeń systemu *Icinga*, komponent ten został podzielony na kilka elementów, które mogą znajdować się na różnych systemach. Można wyróżnić następujące elementy:

IDOMOD — moduł rdzenia monitorującego, który pozwala mu na dostęp do bazy danych.
LOG2IDO — program pozwalający na import utworzonych wcześniej plików do bazy danych.
FILE2SOCK — program pozwalający na przekierowanie danych zapisywanych do pliku do gniazda TCP lub Unix.
IDO2DB — demon, który jest odpowiedzialny za wykonywanie operacji na bazie danych.

Rysunek 4.2. Schemat integracji IDOUtils z systemem Icinga.



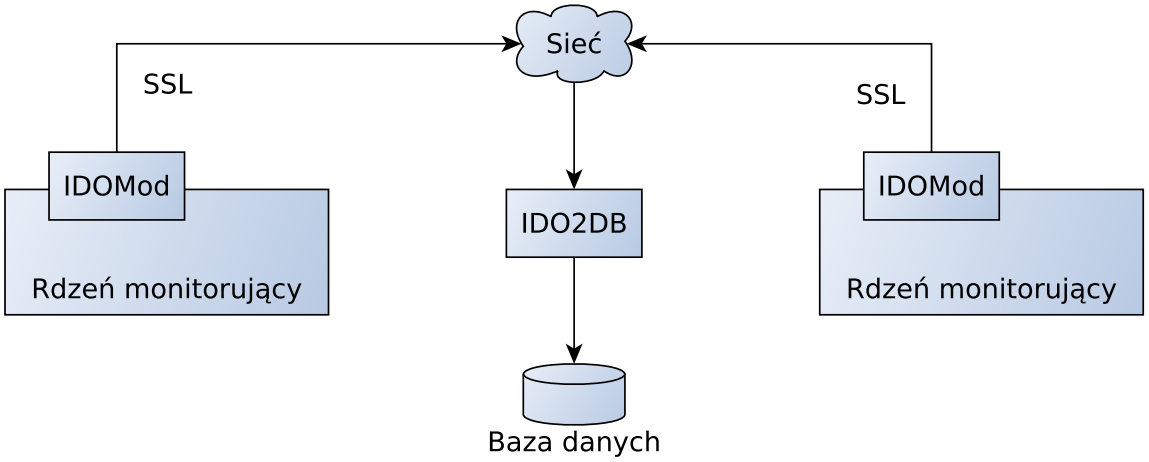
Podstawowymi elementami całego komponentu są IDOMOD oraz IDO2DB. Schemat ich typowego zastosowania przedstawiono na rys. 4.2. Moduł rdzenia IDOMOD ładowany jest przez rdzeń systemu *Icinga* tuż po starcie. Po załadowaniu zapewnia on spójny interfejs do uzyskiwania danych dla wszystkich pozostałych części rdzenia monitorującego. Ponieważ wykonywanie operacji na bazie danych może być czasochłonne, nie powinno być wykonywane przez rdzeń monitorujący. Z tego powodu powstał program IDO2DB. Jest on uruchomiony jako demon na dowolnym urządzeniu. Zadaniem tego serwisu jest fizyczna realizacja żądań na

bazie danych. Na schemacie celowo pominięto pozostałe elementy tego komponentu, gdyż stanowią one jedynie inne źródło danych dla demona IDO2DB i są konieczne jedynie przy imporcie danych historycznych ze starej instalacji systemu, lub bardziej zaawansowanych konfiguracjach.

Ponieważ rdzeń monitorujący oraz demon IDO2DB mogą znajdować się zarówno na jednym komputerze, jak i na różnych konieczne jest zapewnienie odpowiednich mechanizmów komunikacji pomiędzy nimi. Gdy programy te znajdują się na różnych systemach, jako mechanizm komunikacji wykorzystywane są gniazda TCP. W podstawowej konfiguracji dane przekazywane są w sposób niezaszyfrowany. Jeśli jednak istnieje potrzeba zapewnienia tajności oraz integralności przekazywanych danych, możliwe jest użycie protokołu SSL[1]. W sytuacji gdy oba programy uruchomione są na tym samym urządzeniu, w celu poprawy wydajności możliwe jest użycie gniazd protokołu Unix[2].

Program IDO2DB jest bardzo elastyczny. Nie posiada on ograniczenia co do liczby podłączonych do niego modułów IDOMOD czy innych elementów. Pozwala to na wykorzystanie wspólnej bazy danych przez wiele instancji rdzenia monitorującego. Opisana konstrukcja została schematycznie przedstawiona na rys. 4.3. Warto zaznaczyć, że interfejs graficzny icinga-web jest również kompatybilny z tym rozwiązaniem. Pozwala to na wyświetlanie w jednym interfejsie danych pochodzących od wielu rdzeni monitorujących.

Rysunek 4.3. Schemat wykorzystania IDOUtils w systemie Icinga.



W celu zapewnienia możliwości migracji ze środowiska, które korzystało wcześniej z przechowywania danych w plikach, został dostarczony program LOG2IDO. Pozwala on na import danych historycznych do bazy danych. Program ten, analogicznie jak IDOMOD, nie operuje bezpośrednio na bazie danych, lecz komunikuje się tymi samymi metodami, co IDOMOD z demonem IDO2DB. Zarówno program LOG2IDO, jak i moduł IDOMOD mogą kierować żądania do IDO2DB poprzez plik. Aby umożliwić przekazywanie tych danych z pliku do demona IDO2DB, opracowano program FILE2SOCK. Jest to prosty program, który przekazuje dane zapisane do danego pliku do demona IDO2DB. Program ten nie zajmuje się w żadnym

[1] ang. *Secure Socket Layer* – protokół warstwy prezentacji, zapewniający poufność oraz integralność przesyłanych danych.

[2] ang. *Unix Domain Socket* – metoda komunikacji międzyprocesowej w systemach Unix. Posiada jednolite API, jak gniazda domeny internetowej.

stopniu przetwarzaniem odczytaniem danych, lecz jedynie przesłaniem ich poprzez gniazdo internetowe lub Unix do demona IDO2DB.

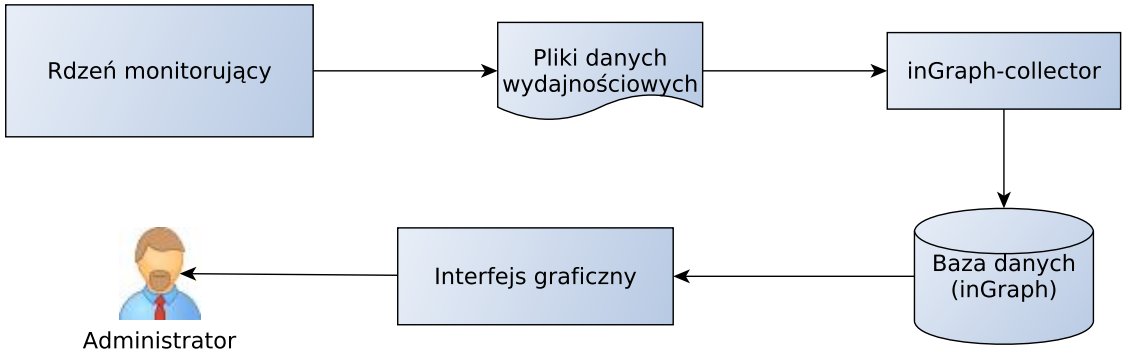
4.3. Dodatek inGraph

inGraph jest dodatkiem do systemów *Icinga* oraz Nagios, który umożliwia prezentację danych zgromadzonych poprzez system monitorujący w postaci wykresów. Dodatek ten został opracowany przez firmę NETWAYS GmbH i wydany na licencji GPL w wersji 3. Cechą, która odróżnia dodatek *inGraph* od innych rozwiązań przeznaczonych do analizy danych historycznych jest wykorzystanie relacyjnej bazy danych do przechowywania danych otrzymanych od systemu monitorującego. Należy podkreślić, ze ten dodatek funkcjonuje jako zupełnie odzielny program. Posiada on ponadto własną, niezależną bazę danych. Jedynym punktem integracji tego dodatku z systemem *Icinga* jest wspólny interfejs graficzny. Na podstawie otrzymanych danych dodatek *inGraph* dokonuje przeliczeń dla odpowiednich przedziałów czasowych, które używane są do późniejszej generacji wykresów. Rozmiary przedziałów definiowane są przez użytkownika w plikach konfiguracyjnych. Wykorzystanie tego rodzaju bazy danych powoduje nieustanny wzrost rozmiaru bazy. W celu optymalizacji zajętości przestrzeni dyskowej dodatek *inGraph* administruje danymi zgodnie z polityką, która również jest definiowana w jego plikach konfiguracyjnych. Dla każdego przedziału czasowego zdefiniowany jest również okres przechowywania danych. Dodatek umożliwia przeglądanie danych dokładnych z najmniejszych przedziałów czasu oraz wykresów długoterminowych prezentujących trendy danej wartości. Ponieważ dane są bezpośrednio administrowane przez dodatek *inGraph*, możliwa jest zmiana czasu przechowywania danych z wskazanych przedziałów nawet w trakcie działania systemu[3].

Dodatek *inGraph* składa się z dwóch niezależnych elementów, komunikujących się poprzez XML-RPC[4]:

ingraph interfejs graficzny,
ingraph-collector program zbierający dane.

Rysunek 4.4. Typowy przepływ danych przy wykorzystaniu dodatku inGraph.



[3] Dla porównania należy przypomnieć systemy oparte na cyklicznych bazach danych, w których rozmiar definiowany może być tylko i wyłącznie podczas tworzenia bazy.

[4] ang. *XML Remote Procedure Call* — zdalne wywołanie procedur przy użyciu XML. Metoda zdalnego wywoływania funkcji oparta na dokumentach w formacie XML. Szczegółowy opis w [17].

Program zbierający oraz przetwarzający dane został napisany w języku Python i nosi nazwę *ingraph-collector*. Jego zadaniem jest pobieranie danych od systemu monitorującego, dokonywanie ich przeliczeń oraz umieszczanie ich wyników w bazie danych.

Do pobierania danych z systemu monitorującego wykorzystano mechanizm udostępniania danych wydajnościowych. Oznacza to, że rdzeń systemu monitorującego przekazuje dane wydajnościowe otrzymane od wtyczek poprzez plik tekstowy. Oczywiście dane muszą być eksportowane przy pomocy formatu zrozumiałego dla dodatku *inGraph*. Demon *ingraph-collector* wykonuje ich analizę, a następnie wykonuje wszystkie niezbędne obliczenia np. średnich wartości w zadanych przedziałach czasowych. Wyniki zapisywane są w bazie danych MySQL lub PostgreSQL programu *inGraph*. Należy zwrócić uwagę, iż jest to inna baza danych, niż ta z której korzysta system *Icinga*. Ważną różnicą pomiędzy danymi składowanymi w tej bazie, a danymi przechowywanymi przez system monitorujący jest ich format. Systemy monitorujące przechowują w postaci numerycznej jedynie skwantowany stan danej usługi lub urządzenia (OK, WARNING itd), pozostałe dane przechowywane są w postaci tekstowej w formie przekazanej przez wtyczkę. Dodatek *inGraph* przechowuje natomiast w swojej bazie dane w postaci już przetworzonej. Oznacza to, iż dokonywany jest rozbiór składniowy rezultatów pomiarów zawartych w danych wydajnościowych i w bazie danych zapamiętywane są pochodzące z tych rezultatów dane w postaci numerycznej. Typowy przepływ danych został przedstawiony na rys. 4.4.

Interfejs użytkownika dodatku *inGraph* został napisany w językach PHP oraz JavaScript. Umożliwia on podgląd danych zebranych i przetworzonych przez rdzeń dodatku. Interfejs może funkcjonować zarówno jako niezależny serwis, jak i integralna część interfejsu systemu *Icinga*. Umożliwia on generację wykresów dla każdego z urządzeń oraz dla każdej z usług. Formaty wykresów, a także przedziały agregacji danych, definiowane są w plikach konfiguracyjnych w formacie JSON[5]. Użytkownik po wybraniu usługi lub urządzenia uzyskuje interaktywny wykres przezentujący dane w zadanym okresie. Wszystkie wykresy wygenerowane przez program są w pełni konfigurowalne i edytowalne. Typ prezentowanych danych jest uzależniony od rozmiaru przedziału czasu, w którym generowany jest wykres.

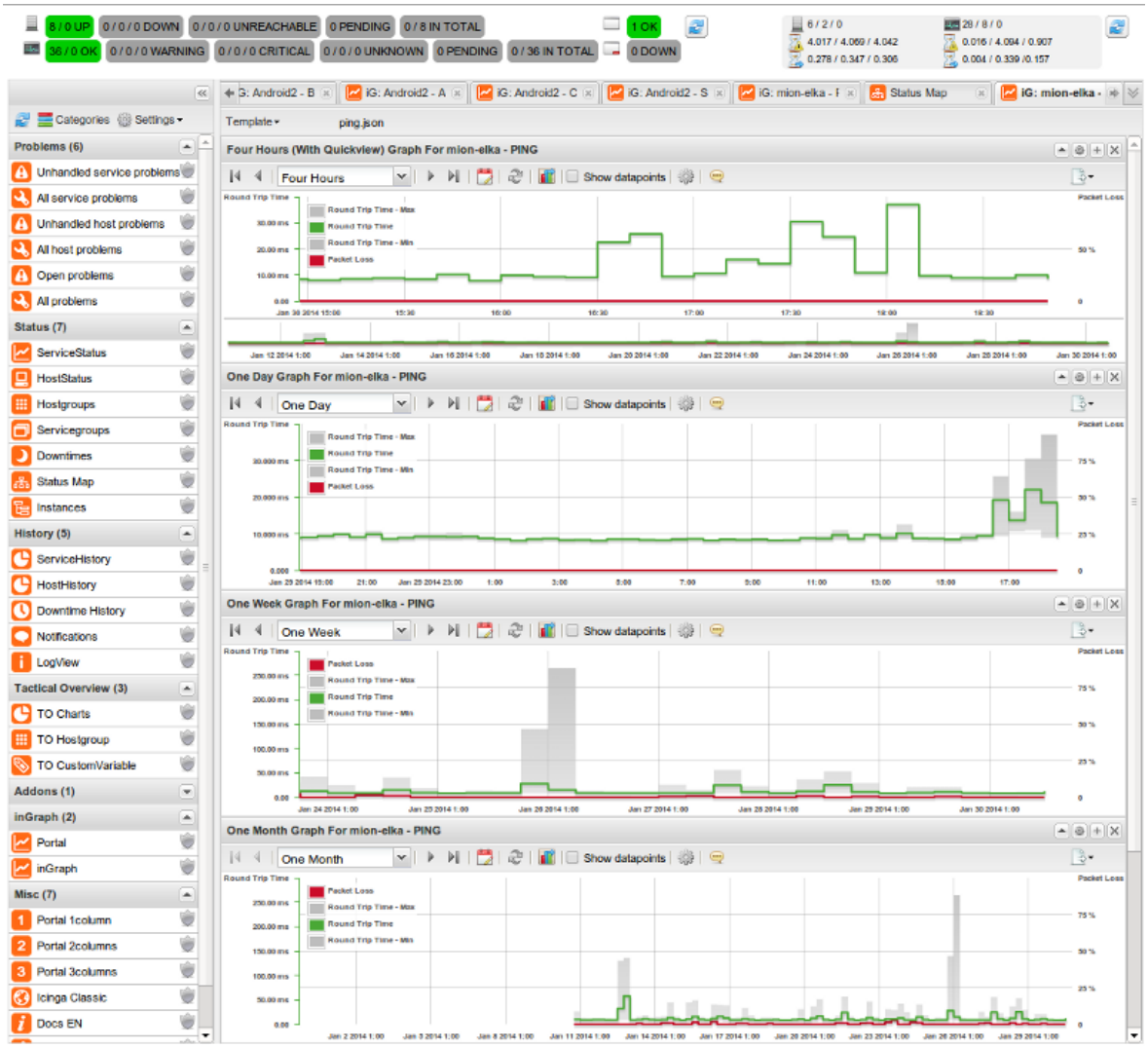
Jeśli okno czasu jest odpowiednio małe, na wykresie zostaną przedstawione dane dokładne. W sytuacji, gdy nie jest możliwe przedstawienie danych dokładnych, ze względu na rozmiar zadanego okresu czasu, dane są agregowane w przedziały, a na wykresie udostępniana jest wartość minimalna, maksymalna oraz średnia dla danego przedziału agregacji danych. Przykładowe wykresy wygenerowane przy pomocy dodatku *inGraph* przedstawia rys. 4.5. Szczególną uwagę warto zwrócić na szare pola reprezentujące minimum oraz maksimum w danym przedziale.

4.4. Dodatek NSCA

NSCA - Nagios Service Check Acceptor jest to dodatek do systemów monitorujących z rodziny Nagios, więc również systemu *Icinga*. Pozwala on na wykorzystanie

[5] ang. *JavaScrip Object Notation* – lekki format tekstowy wymiany danych komputerowych. Szczegółowo opisany w [5].

Rysunek 4.5. Interfejs dodatku inGraph.

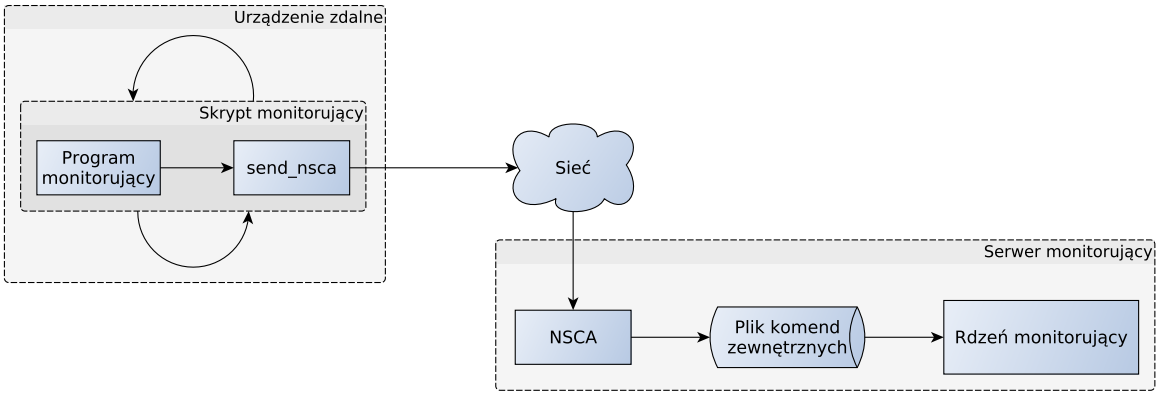


mechanizmów pasywnego monitorowania z systemu innego niż ten, na którym uruchomione jest oprogramowanie monitorujące. Program ten został napisany w całości w języku C i wydany na licencji pozwalającej na wgląd do kodu źródłowego. Wykorzystuje on plik komend zewnętrznych (patrz rozdz. 4.1) i nie integruje się z rdzeniem monitorującym. Dzięki temu możliwe jest jego wykorzystanie go w wielu konfiguracjach bez konieczności ingerencji w system monitorujący.

Schemat działania systemu wykorzystującego dodatek *NSCA* został przedstawiony na rys. 4.6. Implementacja dodatku składa się z dwóch modułów:

— moduł wysyłający (send_nsca) służący do wysyłania wyników sprawdzeń z monitorującego systemu do centralnego serwera, na którym umieszczony jest rdzeń systemu monitorującego odpowiedzialny za przetwarzanie wyników sprawdzeń,
— moduł odbierający (nsca) służący do odbierania wyników sprawdzeń od klientów i dostarczaniu ich do pliku komend zewnętrznych danego systemu monitorującego (patrz rozdz. 4.1).

Rysunek 4.6. Schemat działania dodatku NSCA.



send_nsca

Ta część dodatku jest cyklicznie uruchamiana na systemie, na którym funkcjonuje jakiś mechanizm sprawdzający, który wykonuje sprawdzenia stanu parametru lub usługi. Mechanizm sprawdzający (np. skrypt uruchamiany co 10 minut) wykonuje sprawdzenie aktualnego stanu usługi, a następnie przekazuje go na standardowe wejście programu wysyłającego (send_nsca). Moduł wysyłający, po uruchomieniu odczytuje ustawienia z pliku konfiguracyjnego, a następnie próbuje połączyć się z serwerem (*NSCA*) używając protokołu TCP. Po udanej próbie połączenia otrzymuje pakiet inicjujący, który zawiera:

wektor inicjujący — wygenerowany przez serwer pseudolosowy ciąg znaków konieczny do inicjalizacji algorytmu kryptograficznego,
stempel czasu — czas odczytany przez serwer w chwili nadejścia połączenia od klienta.

Po otrzymaniu pakietu inicjującego moduł wysyłający rozpoczyna czytanie danych ze standardowego wejścia. Wszystkie dane wejściowe muszą być odpowiednio sformatowane: poszczególne pola informacyjne muszą być rozdzielone pojedynczą tabulacją, a cały rezultat pomiaru zakończony znakiem nowej linii. Sprawdzenia dotyczące urządzenia powinny zawierać następujące pola:

nazwa urządzenia — krótka nazwa urządzenia, którego stan jest przekazywany,
stan — numerycznie wyrażony kod stanu urządzenia,
odczyt — dodatkowe wartości odczytów opisujące stan urządzenia w formacie zgodnym z formatem danych przekazywanych przez wtyczki (patrz rozdz. 4.1).

Natomiast wpisy dotyczące usług powinny zawierać następujące pola:

nazwa urządzenia — krótka nazwa urządzenia, na którym uruchomiona jest usługa,
opis usługi — nazwa usługi danego urządzenia, której dotyczy wpis
stan — numerycznie wyrażony kod stanu usługi,
odczyt — dodatkowe wartości odczytów opisujące stan usługi w formacie zgodnym z formatem danych przekazywanych przez wtyczki (patrz rozdz. 4.1).

Łatwo zauważyć, że żadne z pól wpisu dziennika nie zawiera stempla czasu wymaganego przez rdzeń sprawdzający przy zapamiętywaniu odczytu pasywnego. Dzieje się tak, gdyż program *NSCA* posiada zdefiniowaną własną politykę określania

czasu wykonania sprawdzenia. Do każdego pakietu zawierającego wpis dziennika dodawany jest stempel czasu otrzymany w pakiecie inicjującym od modułu odbierającego. Właściwy stempel czasu, który trafia do rdzenia monitorującego nadawany jest natomiast przez moduł odbierający. Oznacza to przekłamania czasu pomiary, przy próbie transmisji danych historycznych, zgromadzonych na skutek np. utraty łączności.

Kolejnym krokiem działania modułu jest obliczenie cyklicznego kodu nadmiarowego CRC32 dla danego pakietu. Po dołączeniu obliczonego kodu do pakietu pakiet jest szyfrowany. Algorytm kryptograficzny stosowany do szyfrowania pakietów został wcześniej zainicjalizowany wektorem pseudolosowych danych odebranych w pakiecie inicjalizacyjnym od modułu odbierającego. Po zaszyfrowaniu dane są wysyłane, a moduł wysyłający, bez oczekiwania na potwierdzenie przetworzenia przez serwer, rozpoczyna przetwarzanie kolejnej porcji danych. Opis przetwarzania pakietu po stronie serwera *NSCA* został opisany dalej w tym rozdziale.

Moduł odbierający

Demon, który stanowi moduł odbierający funkcjonuje na tym samym systemie operacyjnym, na którym znajduje się rdzeń systemu monitorującego. Ta część odpowiedzialna jest za odbieranie danych od klientów i przekazywanie ich do rdzenia programu monitorującego. Moduł ten może pracować w jednym z poniższych trybów:

samodzielny demon jednoprocesowy — uruchomiony w tle demon, który nasłuchuje na przychodzące połączenia od klientów i po nadejściu połączenia jest ono obsługiwane przy użyciu jednego procesu z jednym wątkiem,

samodzielny demon wieloprocesowy — uruchomiony w tle demon, którego proces główny nasłuchuje na nadejście połączeń od klientów, gdy takie połączenie nadejdzie proces jest duplikowany i każdy z klientów obsługiwany jest w innym procesie potomnym,

demon zintegrowany z inetd — w systemie uruchomiony jest demon inetd, który nasłuchuje na połączenia od klientów na konkretnym gnieździe, a gdy nadejdzie połączenie od klienta uruchamiany jest proces demona *NSCA*, który obsługuje nowe połączenie i kończy się wraz z zakończeniem obsługi klienta.

Niezależnie od trybu pracy, do przekazywania odebranych od zdalnego systemu danych do rdzenia monitorującego używany jest mechanizm pasywnego monitorowania dostępny w systemach z rodziny Nagios. Aby możliwe było wykorzystanie tego mechanizmu do przekazania danych z innych urządzeń konieczne jest zapewnienie demonowi *NSCA* dostępu do pliku zewnętrznych komend systemu monitorującego (patrz rozdz. 4.1). Ponieważ plik ten jest potokiem nazwanym, chroniony jest on przez Uniksowy system uprawnień użytkowników. Zapewnienie dostępu do takiego bytu może się odbyć na dwa sposoby. Pierwszym, polecanym przez twórców systemów monitorujących, jest uruchamianie demona *NSCA* jako procesu tego samego użytkownika co proces rdzenia systemu monitorującego. Drugim sposobem jest modyfikacja praw dostępu do omawianego pliku, tak aby umożliwić dostęp użytkownikowi, z którego uprawnieniami uruchomiony jest demon *NSCA*. Przy zastosowaniu drugiego rozwiązania zalecana jest szczególna ostrożność, gdyż dostęp do pliku komend zewnętrznych daje bardzo duże możliwości ingerencji w system monitorujący.

Komunikacja modułu odbierającego z klientem rozpoczyna się od nadejścia połączenia od klienta. Gdy moduł odbierający otrzyma nowe połączenie zostanie wysłany pakiet inicjalizujący, którego zawartość została opisana już wcześniej w tym rozdziale. Po przesłaniu pakietu inicjalizującego połączenie, moduł odbierający oczekuje na dane od klienta. Każdy wynik sprawdzenia przesyłany jest przy użyciu pakietu o poniższych polach:

wersja protokołu — aktualnie używana wersja protokołu komunikacyjnego,
kod CRC32 — kod CRC32 bieżącego pakietu,
stempel czasu — stempel czasu pochodzący z pakietu inicjalizującego przesłanego klientowi,
kod statusu — kod stanu usługi/hosta powiązany z przesyłanym wpisem
nazwa hosta — nazwa urządzenia, które podlegał sprawdzeniu. Nie jest konieczne aby było to to samo urządzenie, który dostarcza dane,
opis usługi — nazwa usługi, która podlegała sprawdzeniu lub pusty napis jeśli sprawdzenie dotyczy urządzenia,
wynik sprawdzenia — napis wygenerowany przez wtyczkę, która dokonywała sprawdzenia, zawierający dodatkowe dane na temat stanu urządzenia lub usługi.

Pakiety są zaszyfrowane z użyciem algorytmu oraz klucza symetrycznego pochodzącego z pliku konfiguracyjnego demona *NSCA*. Po odebraniu spodziewanej ilości danych (wszystkie pakiety mają taką samą długość wynikającą z rozmiaru struktury używanej w programie), następuje próba odszyfrowania odebranych danych. Sprawdzenie poprawności odebranych danych i jednocześnie weryfikacja uprawnień odbywa się poprzez kontrolę zawartości pola CRC32. Jeśli wartość znajdująca się w tym polu zgadza się z wartością wyliczoną dla otrzymanych danych, to pakiet jest przyjmowany, w przeciwnym zaś razie pakiet zostanie odrzucony bez powiadamiania o tym jego nadawcy. Dalsze przetwarzanie otrzymanego pakietu rozpoczyna się od porównania bieżącego stempla czasu z tym pochodzącym z odebranego pakietu. Jeśli różnica pomiędzy nimi jest zbyt duża, dane zostają odrzucone. Ostatnią czynnością wykonywaną przez moduł odbierający jest zapisanie odebranego wpisu do pliku komend zewnętrznych systemu monitorującego.

Warto wspomnieć, że stempel czasu przesłany przez klienta nie jest dostarczany do jądra monitorującego. Służy on jedynie określeniu odstępu czasu od inicjalizacji sesji do chwili otrzymania wiadomości i podjęciu decyzji o przyjęciu, bądź odrzuceniu pakietu. Do systemu monitorującego trafia natomiast bieżący stempel czasu serwera, na którym uruchomiony jest moduł odbierający i jądro systemu monitorującego. Do generacji stempla czasu wykorzystywany jest czas uniwersalny. W związku z powyższym dodatek *NSCA* nie umożliwia synchronizacji danych historycznych. Wszystkie dane, które nadejdą są traktowane tak, jakby pomiary były wykonane właśnie w tej chwili.

Istotną może się również okazać informacja, iż protokół komunikacyjny nie przewiduje przesyłania ACK[6], bądź też NACK[7]. Moduł wysyłający ma zatem pewność, iż wysłane przez niego dane zostaną dostarczone, gdyż używany jest protokół TCP.

[6] ang. *Acknowledgement* – pozytywne potwierdzenie, powszechnie przyjęta nazwa komunikatu potwierdzającego przyjęcie i przetworzenie danych przez aplikację.

[7] ang. *Negative Acknowledgement* – potwierdzenie negatywne, powszechnie przyjęta nazwa komunikatu oznaczająca odmowę przyjęcia lub przetworzenia odebranych danych.

Nie ma jednak żadnej gwarancji ani informacji, że dane przesłane do modułu odbierającego zostaną dostarczone do rdzenia systemu monitorującego.

Bezpieczeństwo monitorowania z użyciem dodatku *NSCA* opiera się na kryptografii symetrycznej oraz cyklicznym kodzie nadmiarowym CRC32 (wielomian: 0xEDB88320). Wiadomość inicjująca połączenie jest nieszyfrowana. Natomiast każda wiadomość zawierająca dane pomiarowe jest zaszyfrowana algorytmem wybranym podczas konfiguracji systemu. Dodatek *NSCA* korzysta z biblioteki *libmcrypt*[8] i umożliwia użycie jednego spośród wielu algorytmów kryptografii symetrycznej, które zostały w niej zaimplementowane. Użytkownik posiada jedynie możliwość wyboru stosowanego algorytmu, natomiast jako tryb pracy stosowany jest tryb sprzężenia zwrotnego szyfrogramu (CFB — ang. *Cipher Feedback*). Tryb ten wymaga zawsze inicjalizacji zarówno kodera jak i dekodera tym samym wektorem początkowym, który w przypadku tego protokołu, jest przesyłany przez serwer w pakiecie inicjującym.

Wszystkie algorytmy symetryczne do prawidłowego działania wymagają, aby komunikujące się strony współdzieliły pewien sekret jakim jest klucz używany do szyfrowania. Ujawnienie klucza symetrycznego wiąże się z kompromitacją całego systemu kryptograficznego. W dodatku *NSCA* klucz ten uzyskiwany jest z hasła, które musi być zapisane przez administratora systemu w pliku konfiguracyjnym zarówno części odbierającej, jak i wysyłającej. Oczywistym jest, iż poza współdzieleniem klucza, wszystkie komunikujące się węzły muszą używać tego samego algorytmu kryptograficznego.

Algorytmy szyfrowania zapewniają tajność przesyłanej wiadomości, jednak w przypadku systemu monitorowania potrzebne jest również zapewnienie integralności. Integralność w dodatku *NSCA* zapewniana jest poprzez cykliczny kod nadmiarowy CRC32. Przed zaszyfrowaniem wiadomości obliczany jest jej kod CRC, który jest dołączany do wiadomości. Następnie wiadomość jest szyfrowana i przesyłana do serwera *NSCA*. Po odebraniu wiadomości jest ona odszyfrowywana i następuje weryfikacja kodu CRC. Jeśli weryfikacja się nie powiedzie pakiet jest oznaczany jako dane z naruszoną integralnością i w konsekwencji odrzucany bez powiadomienia o tym klienta. W szczególności, taka sytuacja może się zdarzyć, gdy klient używa innego algorytmu kryptograficznego lub klucza. Pakiety, których integralność nie zostanie pozytywnie zweryfikowana są odrzucane.

Model bezpieczeństwa zastosowany w dodatku *NSCA* ma kilka wad. Warto przypomnieć, iż wszystkie ustawienia zarówno modułu wysyłającego jak i odbierającego przechowywane są w plikach na dyskach odpowiednich urządzeń. Pliki te zawierają również klucze symetryczne, które są stosowane w całym systemie. Oznacza to, iż uzyskanie dostępu typu odczyt do takiego pliku powoduje utratę tajności danych przesyłanych w całym systemie. Ponadto przyjęty model bezpieczeństwa, nie zawiera żadnej weryfikacji danych pochodzących od klientów. Oznacza to, że każdy klient może przesłać wpisy dziennika, udające wpisy pochodzące od zupełnie innych klientów. W szczególności, jeśli atakujący uzyska klucz symetryczny, to nie tylko będzie mógł odczytywać informacje o danych przesyłanych od klientów, lecz także podszywać się pod klientów i przesyłać fałszywe wpisy. Taka luka może być wykorzystana przy ataku na jakąś usługę lub urządzenie. Atakujący rozpoczyna atak, po czym przechwytuje pakiety z wpisami dziennika, które mogą świadczyć

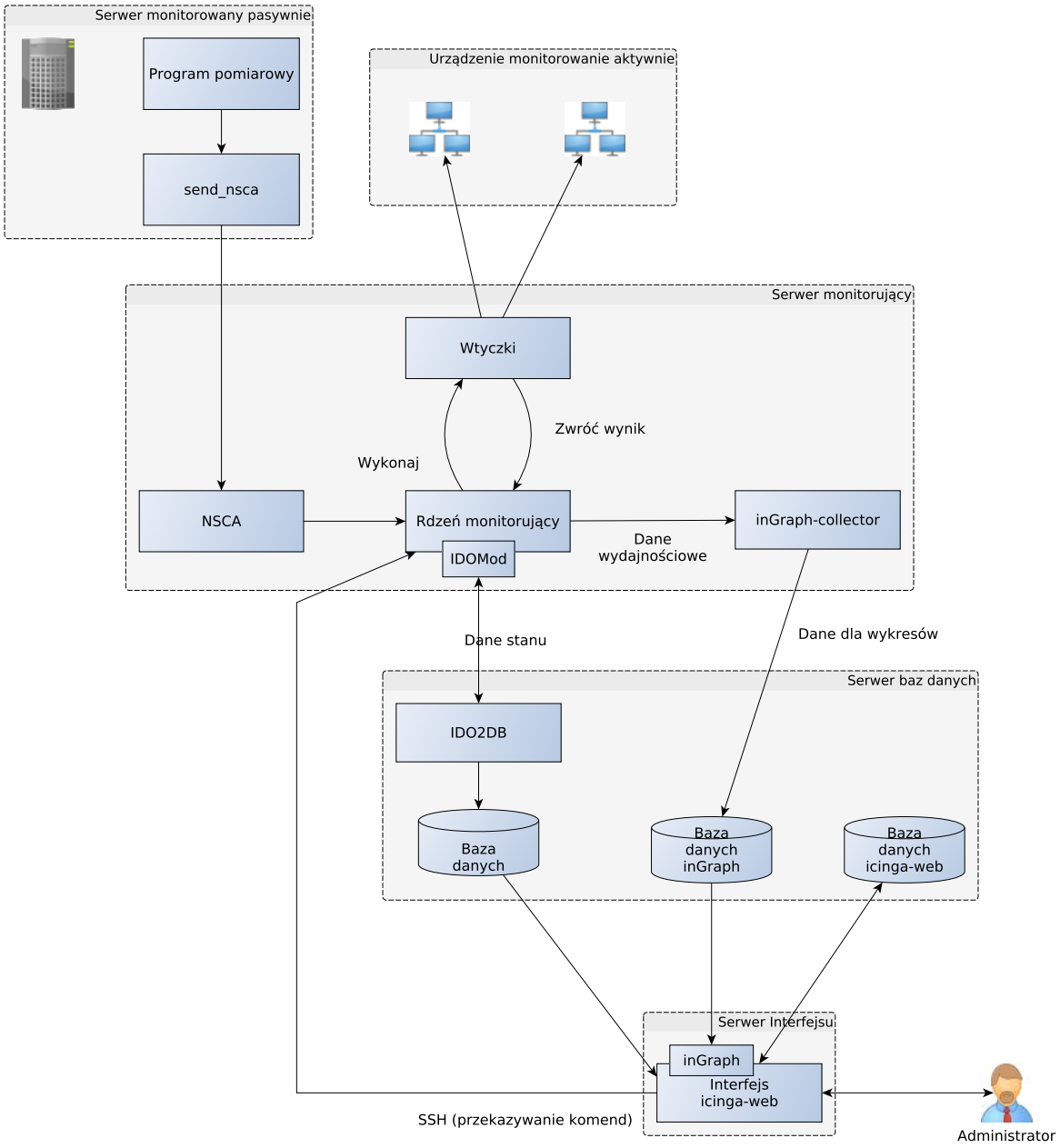
[8] Szczegółowy opis biblioteki jak i dostępnych w niej algorytmów można znaleźć w [6].

o rozpoczęciu ataku i w zamian przesyła do serwera fałszywe pakiety informujące, że wszystkie usługi pracują normalnie.

4.5. Podstawowe konfiguracje rozproszone

Przedstawione w trakcie tego rozdziału elementy pozwalają na zbudowanie w pełni funkcjonalnego hybrydowego systemu monitorowania. Schemat takiego systemu został przedstawiony na rys. 4.7.

Rysunek 4.7. Schemat podstawowej konfiguracji systemu Icinga.



Konfiguracja systemu podzielona jest na kilka części, z których każda może znajdować się na innym urządzeniu. Podstawowym elementem systemu jest serwer monitorujący. Zawiera on przede wszystkim rdzeń monitorujący *Icinga* oraz wtyczki niezbędne do monitorowania infrastruktury IT. Aby umożliwić monitorowanie urządzeń w sposób pasywny na tym samym urządzeniu umieszczony może być program *NSCA*, który będzie przekazywał dane pochodzące z urządzeń monitorowanych pasywnie do rdzenia poprzez plik komend zewnętrznych. Ponadto na tym samym serwerze musi znajdować się *ingraph-collector*[9]. Demon ten uzyskuje dane poprzez eksportowane przez rdzeń monitorujący pliki z danymi wydajnościowymi więc nie jest możliwe umieszczenie go na innym urządzeniu.

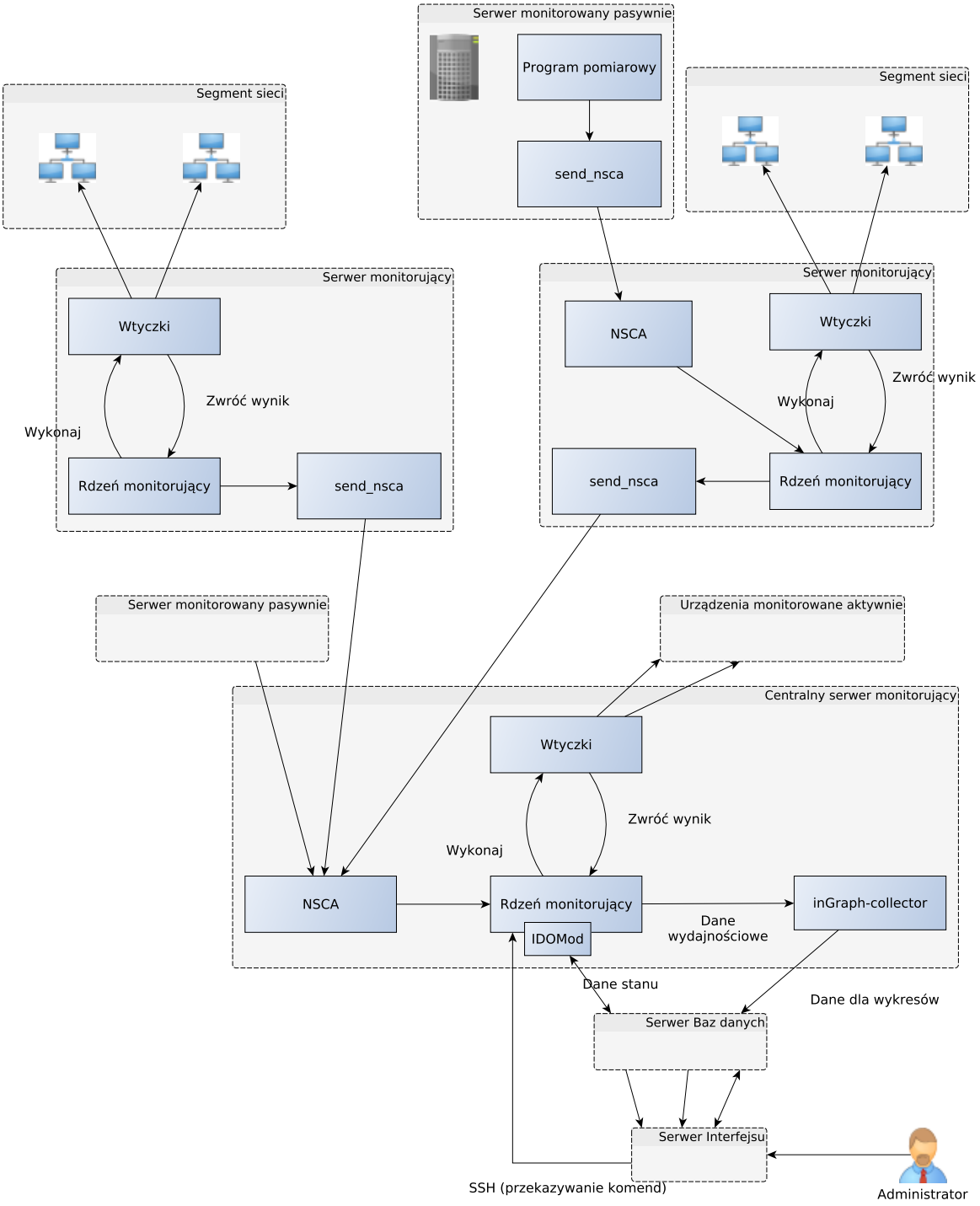
Kolejnym elementem systemu jest serwer baz danych. Zawiera on trzy bazy danych: systemu *Icinga*, dodatku *inGraph* oraz interfejsu graficznego. Baza danych systemu *Icinga* wypełniana jest poprzez komponent *IDOUtils*, który umieszcza tam dane pochodzące z rdzenia monitorującego. Baza ta jest niezbędna do prawidłowego funkcjonowania interfejsu graficznego, gdyż jest ona dla niego źródłem danych o stanie infrastruktury. Baza danych dodatku *inGraph* wypełniana jest przez program *ingraph-collector* na podstawie danych wydajnościowych otrzymanych od rdzenia. W bazie tej przechowywane są wstępnie przetworzone dane konieczne do generacji wykresów. Trzecia baza danych zawiera dane konfiguracyjne interfejsu graficznego, aktualny układ graficzny, konta użytkowników oraz wszystkie inne dane wymagane do poprawnego działania serwisu. W prezentowanej konfiguracji wszystkie bazy danych znalazły się na jednym urządzeniu jednak nie jest to konieczne. Możliwe jest rozmieszczenie każdej z baz na innym urządzeniu. Ponadto bazy te nie są ze sobą w żaden sposób powiązane i każda z nich może posiadać własny format.

Ostatnim elementem systemu jest serwer interfejsu graficznego. Znajduje się na nim interfejs graficzny icinga-web oraz zintegrowany z nim interfejs dodatku *inGraph*. Aby umożliwić funkcjonowanie tego interfejsu, konieczne jest umieszczenie na tym urządzeniu również serwera HTTP jak na przykład Apache. Serwer ten nie został ujęty na rysunku, gdyż nie stanowi logicznego elementu systemu monitorującego. Interfejs graficzny uzyskuje wszystkie niezbędne dane od serwera baz danych. Pozwala on również administratorowi na przekazywanie do rdzenia monitorującego komend zmieniających jego zachowanie. Aby zapewnić taką funkcjonalność, konieczne jest zapewnienie połączenia SSH pomiędzy serwerem interfejsu graficznego, a serwerem monitorującym.

Przedstawiona konfiguracja pozwala na kompleksowe monitorowanie jednosegmentowej sieci. Znaczna cześć przedsiębiorstw posiada jednak sieć składającą się z wielu odrębnych segmentów. Monitorowanie takiej infrastruktury wymaga modyfikacji przedstawionej konfiguracji. Najprostszą metodą jest użycie monitorowania pasywnego dla wszystkich urządzeń nie widocznych z sieci, w której pracuje rdzeń monitorujący. Jest to jednak niewygodne i niezalecane, gdyż nie pozwala w pełni wykorzystać funkcjonalności systemu monitorującego, a także utrudnia konfiguracje. Konieczne jest zatem użycie jednej z dozwolonych konfiguracji zawierających wiele serwerów monitorujących. Jedna z dostępnych konfiguracji wykonanych z użyciem narzędzia *NSCA* została przedstawiona na rys. 4.8.

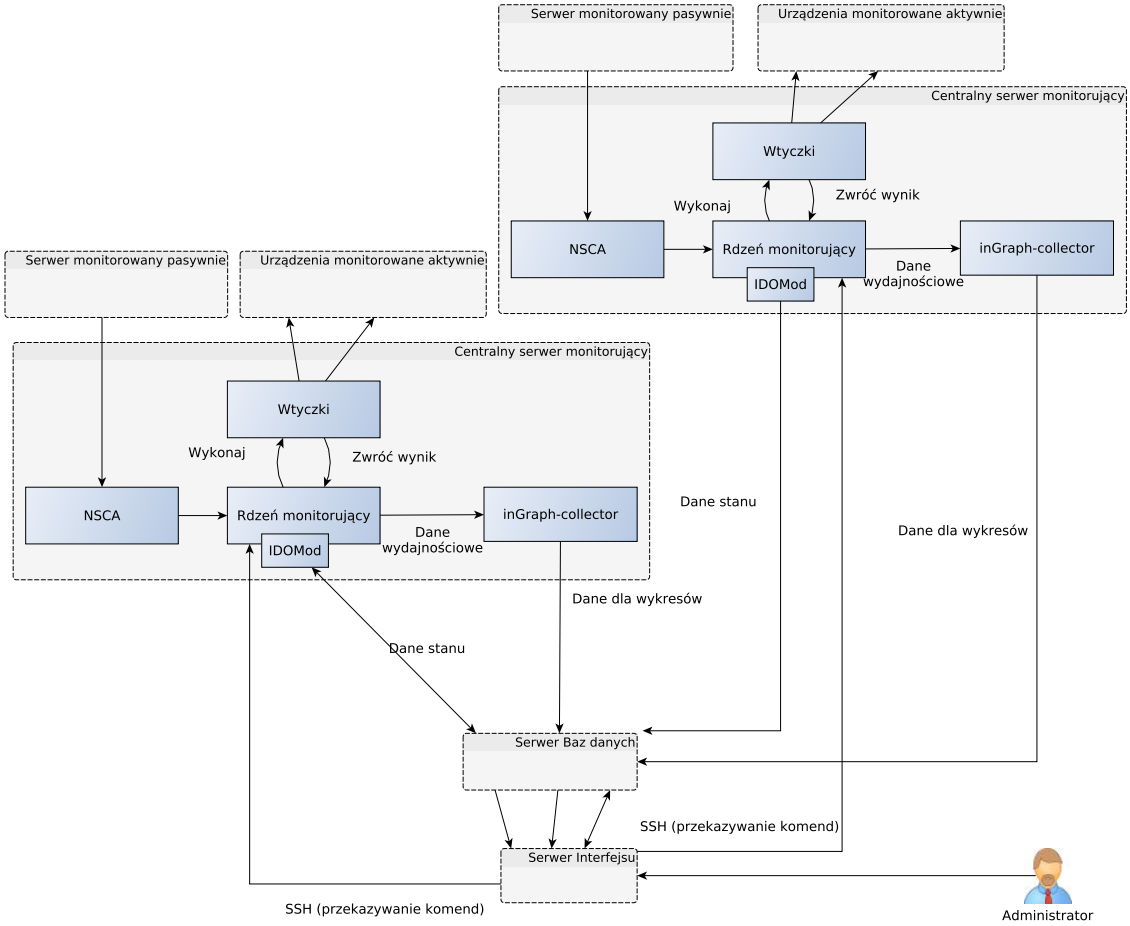
[9] Oczywiście można zrezygnować z tego dodatku, jednak uniemożliwi to analizę danych historycznych oraz generację wykresów.

Rysunek 4.8. Schemat konfiguracji rozproszonej z instancją centralną.



Konfiguracja ta zakłada istnienie jednej instancji centralnej. Instancja ta posiada elementy identyczne jak te, omówione w poprzedniej konfiguracji. Istotną różnicą w stosunku do wcześniej przedstawionej konfiguracji jest istnienie dodatkowych instancji podrzędnych. Typowo, co najmniej jedna instancja umieszczona jest w każdym odizolowanym segmencie sieci. Każdy taki serwer monitorujący może monitorować dany segment zarówno w sposób aktywny, jak i pasywny. Istotne jest natomiast, że dane pochodzące z obu typów monitorowania są przekazywane do instancji centralnej jako usługi monitorowane pasywnie. Oznacza to, że z punktu widzenia instancji centralnej wszystkie usługi są monitorowane w sposób pasywny zatem zajmuje się ona przetwarzaniem wszystkich otrzymanych danych. Jeśli sieć jest bardzo rozległa może to prowadzić do znacznego obciążenia serwera centralnego. Z drugiej strony, dzięki wykonywaniu całego przetwarzania na jednym serwerze możliwa jest bardzo prosta konfiguracja dodatku *inGraph*. Program zbierający dane jest uruchomiony jedynie na centralnej instancji serwera monitorującego gdyż to ona przetwarza wszystkie dane.

Rysunek 4.9. Schemat konfiguracji rozproszonej ze wspólną bazą danych.



Kolejna z konfiguracji rozproszonych zakłada wykorzystanie wspólnej bazy danych. Istnieje wiele instancji systemu monitorującego. Każda z nich jest niezależne i może być nazywana instancją centralną. Każdy serwer monitoruje wyznaczony segment lub grupę urządzeń. Przetworzone dane o stanie usług przekazywane są do wspólnej bazy danych programu *Icinga*. Interfejs graficzny odczytuje te dane z bazy

i prezentuje użytkownikowi globalny stan całej infrastruktury. Każdy z serwerów odpowiedzialny jest za przetwarzanie danych zebranych z wyniku monitorowania wyznaczonej grupy urządzeń. Oznacza to, że każdy z rdzeni monitorujących eksportuje na swoim serwerze dane wydajnościowe pochodzące od monitorowanych urządzeń. Wymusza to konieczność istnienia na każdym z serwerów programu *ingraph-collector*, który będzie przetwarzał dane od wyznaczonego rdzenia monitorującego. Należy zwrócić uwagę, że podczas konfiguracji systemu konieczne jest zadbanie o unikalne nazwy urządzeń w ramach całej infrastruktury, gdyż program ingraph nie posiada żadnych mechanizmów wykrywania takiej sytuacji i różne instancje *ingraph-collector* mogłyby wzajemnie nadpisywać gromadzone dane.

Należy zwrócić uwagę, że system *Icinga* pozwala na dowolne zagnieżdżanie przedstawionych konfiguracji. Oznacza to, że możliwe jest zbudowanie całego drzewa zależności i przekazywania danych pomiędzy poszczególnymi instancjami.

Oba rozwiązania posiadają zarówno zalety jak i wady. Rozwiązanie z użyciem dodatku *NSCA* zapewnia spójne przetwarzanie danych przez jedna instancję i łatwość konfiguracji dodatków wykorzystujących dane eksportowane przez jądro w postaci danych wydajnościowych. Niestety rozwiązanie to generuje znaczące obciążenie instancji centralnej, gdyż musi ona przetwarzać wszystkie wyniki sprawdzeń. Ponadto należy przypomnieć, że dodatek *NSCA* nie posiada mechanizmu potwierdzającego przekazanie danych do rdzenia monitorującego. Oznacza to, że w przypadku niewłaściwej konfiguracji dane mogą być gubione bez powiadamiania o tym użytkownika. Rozwiązanie oparte o wspólną bazę danych posiada rozproszony mechanizm przetwarzania sprawdzeń jak i zdarzeń, dzięki czemu nie występuje w nim nadmierne obciążenie jednej z instancji. Ponadto awaria, dowolnej z instancji nie powoduje nigdy braku możliwości monitorowania całej sieci lecz jedynie jej fragmentu. Niestety w rozwiązaniu tym konieczna jest bardziej zaawansowana konfiguracja dodatków korzystających z danych wydajnościowych. Wybór konfiguracji zależy zatem silnie od infrastruktury w jakiej ma być ona zastosowana, a także od pozostałych elementów systemu, jakie będą wykorzystane.

5. Koncepcja rozwiązania

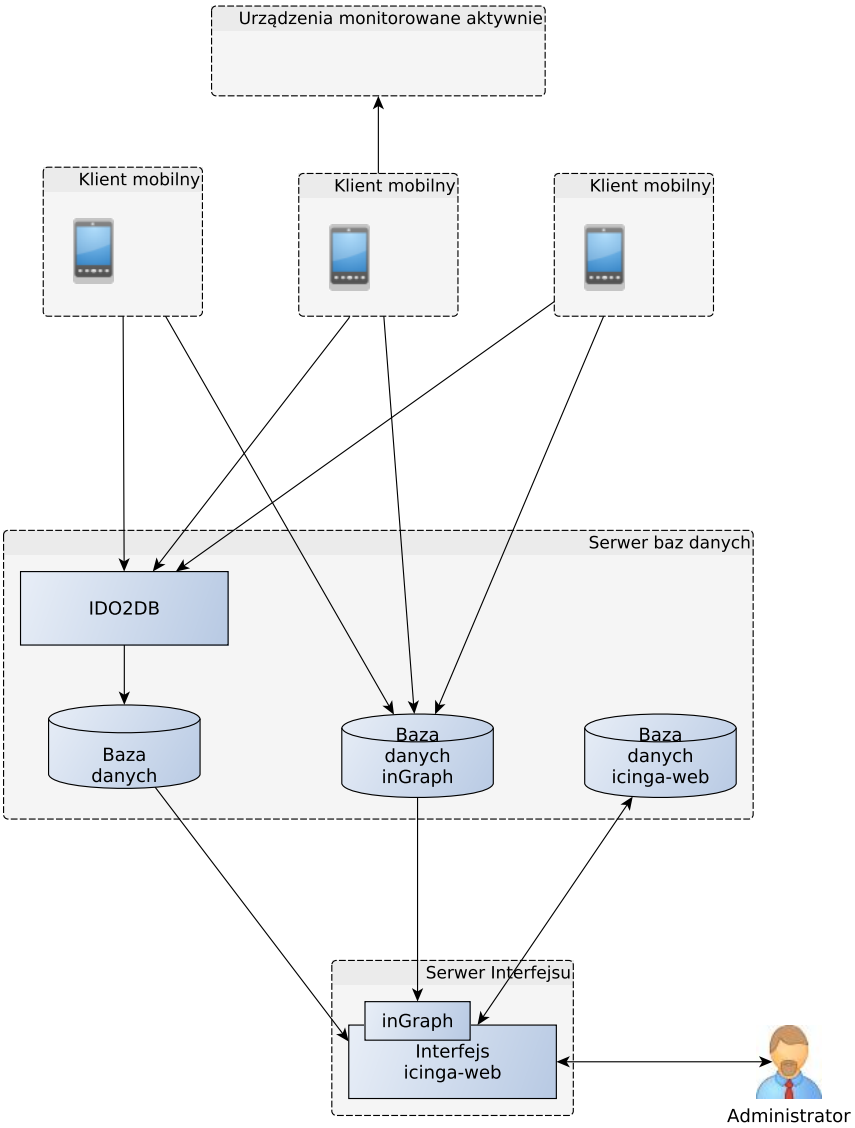
W rozdziale 4 omówione zostały poszczególne komponenty systemu *Icinga*, a także w pełni funkcjonalna konfiguracja dla pojedynczego segmentu sieci. Przedstawiono również dostępne w systemie *Icinga* schematy współpracy wielu serwerów monitorujących. Problem monitoringu klienta mobilnego jest w wielu aspektach symetryczny względem monitoringu wielu segmentów sieci. W tym rozdziale zawarto porównanie tych zagadnień oraz przedstawiono możliwe architektury rozwiązania. Na podstawie wymagań oraz znajomości cech każdej z nich wybrano konfigurację spełniającą w największym stopniu wymagania stawiane przed systemem monitorowania klienta mobilnego. Wybrana architektura opiera się na wykorzystaniu narzędzia NSCA. Niestety narzędzie to nie spełnia wielu spośród stawianych przed nim wymagań. Stwarza to konieczność opracowania nowego dodatku do systemu *Icinga*. W rozdziale tym zawarto konceptualny opis architektury proponowanego dodatku. Ponadto zaproponowano protokół komunikacyjny przeznaczony do przesyłania wyników pomiarów poprzez sieć publiczną.

Przedstawiona w tym rozdziale koncepcja systemu jest elastyczna i zapewnia dobrą skalowalność. Podczas projektowania tego rozwiązania, dla uproszczenia, celowo pomijano zagadnienie monitorowania klienta statycznego. Zgodnie z przedstawionym w rozdziale 4 opisem systemu *Icinga* oraz dostępnych jego konfiguracji rozproszonych, istnieje dowolność przy zagnieżdżaniu konfiguracji systemu. Oznacza to, iż system monitorowania klienta mobilnego może być zintegrowany z każdą infrastrukturą monitorującą opartą na systemie *Icinga*. Spośród wszystkich dostępnych możliwości konfiguracji zalecane jest użycie tej, wykorzystującej wspólną bazę danych zarówno dla klientów statycznych jak i mobilnych, gdyż cechuje się ona najlepszą skalowalnością.

5.1. Architektura systemu

Charakterystyka problemu monitorowania klienta mobilnego przedstawiona w rozdziale 3 wydaje się być w wielu aspektach zbieżna z zagadnieniem monitorowania wielosegmentowej infrastruktury statycznej. Każdy klient mobilny (lub grupa, w ramach której ściśle on funkcjonuje np. zestaw laptop + telefon) może być rozważana jako osobny segment sieci. Oznacza to, że na każdym z klientów mobilnych powinien znajdować się serwer monitorujący. Aby wykorzystać możliwie najwięcej z dostępnych funkcjonalności systemu monitorującego *Icinga* należy zatem rozważyć dostępne architektury rozproszone w kontekście monitorowania klienta mobilnego. System *Icinga* udostępnia dwie zasadnicze konfiguracje rozproszone, które zostały przedstawione w rozdziale 4. Wybór konfiguracji odpowiedniej dla monitorowania klienta mobilnego wymaga analizy obu dostępnych konfiguracji. Jako pierwsza została rozważona konfiguracja ze wspólną bazą danych.

Rysunek 5.1. Monitoring klienta mobilnego w konfiguracji ze wspólną bazą danych.



Architektura pozwalająca na współpracę wielu instancji poprzez wspólną bazę danych została przedstawiona na rys. 5.1. W tej konfiguracji każdy klient mobilny posiada swoją instancję rdzenia monitorującego systemu *Icinga*. Komunikuje się on bezpośrednio z serwerem baz danych, gdzie przechowywany jest stan bieżący całej infrastruktury. Oznacza to konieczność udostępnienia baz danych w sieci publicznej, co zmniejsza poziom bezpieczeństwa systemu. Należy zwrócić uwagę, że każdy klient mobilny jest w pełni odpowiedzialny za przetwarzanie wyników pomiarów wszystkich monitorowanych przez niego urządzeń. W szczególności oznacza to, że implementacja klienta mobilnego musi być wyposażona w mechanizm przetwarzania danych zgodny z dodatkiem inGraph. Konieczne jest również zapewnienie możliwości użycia innego mechanizmu opartego o przetwarzanie danych wydajnościowych, aby nie ograniczać modyfikowalności systemu. Należy zwrócić uwagę, że to rozwiązanie wymaga niemalże ciągłego kontaktu z bazą danych lub opracowania algorytmu synchronizacji danych pomiędzy lokalną kopią przetworzonych danych

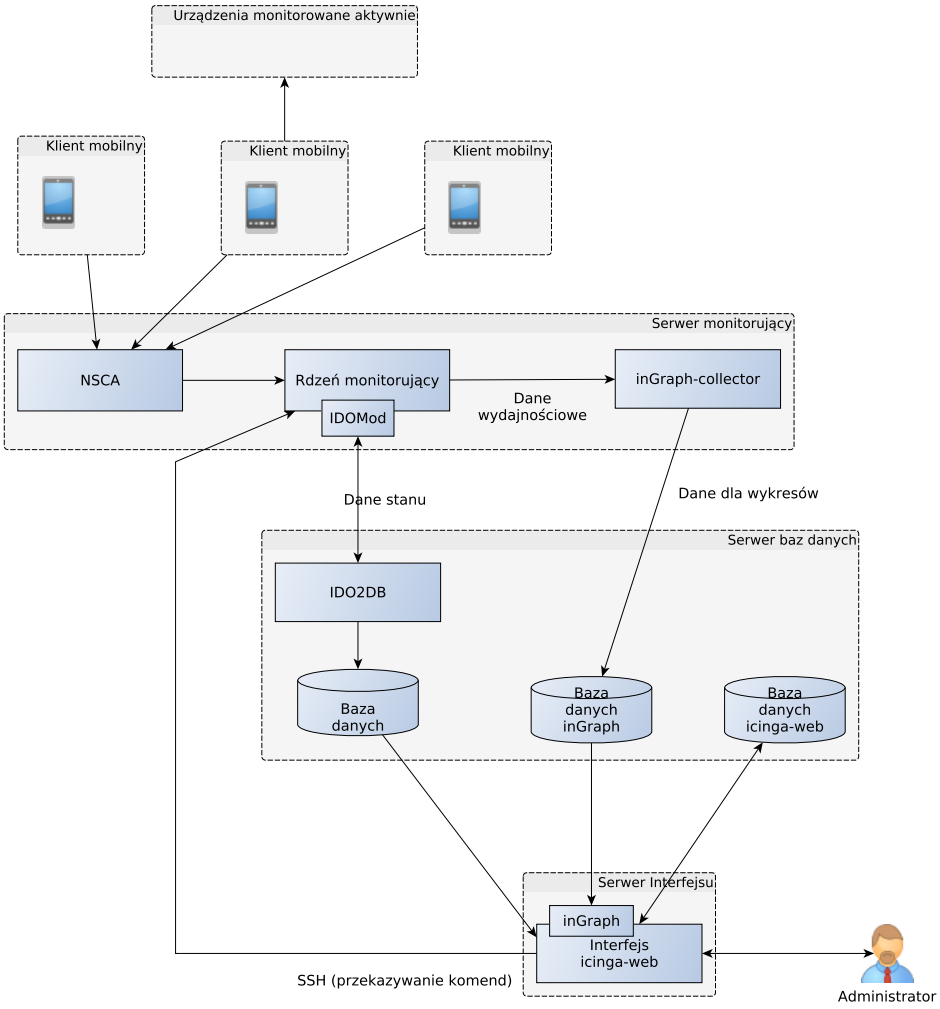
u klienta i baza danych na serwerze. Proces synchronizacji tych danych może wymagać przesłania relatywnie dużej ilości informacji poprzez sieć. Należy również zaznaczyć, że całe przetwarzanie danych, zarówno to związane z monitorowaniem, jak i to związane z wykonywaniem transformacji danych potrzebnych do generacji wykresów jest wykonywane na urządzeniu mobilnym.

Łatwo zauważyć, że konfiguracja ta jest zdecydowanie niewskazana dla klienta mobilnego. Zastosowanie jej wymagałoby znacznego obciążenia urządzenia mobilnego poprzez angażowanie go w przetwarzanie danych. Ponadto rozwiązanie to może zużywać dużą część pasma w zależności od zastosowanego algorytmu synchronizacji danych. Znaczną część przesyłanych danych stanowią przetworzone dane niezbędne do generacji wykresów. Istotny jest również fakt, że jeśli klient mobilny przetwarza dane to jest on również odpowiedzialny za powiadamianie administratora o zdefiniowanych wydarzeniach. Wysyłanie maili zawierających informacje o błędzie generuje oczywiście dodatkowy ruch i obciąża system urządzenia. Warto również wspomnieć o konieczności udostępnienia publicznie bazy danych zarówno systemu monitorującego, jak i dodatku inGraph. Obie te bazy danych mogą zawierać wrażliwe dane przedsiębiorstwa i nie powinny być udostępniane poza sieć prywatną. Wszystkie te elementy stoją w sprzeczności z wymaganiami zdefiniowanymi w 3. W związku z powyższym nie jest możliwe zastosowanie tej architektury do monitorowania klienta mobilnego.

Druga z rozważanych konfiguracji rozproszonych zakłada wykorzystanie dodatku NSCA (rys. 5.2). W tej konfiguracji klient mobilny musi posiadać jedynie prosty mechanizm pozwalający na cykliczne uruchamianie sprawdzeń odpowiednich parametrów i usług. Ponadto warto rozważyć wzbogacenie klienta mobilnego o funkcjonalność monitorowania pasywnego innych urządzeń powiązanych z nim (np. laptop, telefon itd). Można zatem stwierdzić, iż jest to minimalna implementacja rdzenia monitorującego. Zadaniem klienta mobilnego jest jedynie samo sprawdzenie stanu poszczególnych usług, zgodnie z polityką zdefiniowaną przez administratora. Dane zebrane w ramach monitorowania muszą być przechowane na urządzeniu i gdy stanie się to możliwe dostarczone do serwera NSCA jako dane monitorowania pasywnego. Rdzeń monitorujący umieszczony na serwerze po odebraniu danych dokona ich przetworzenia zarówno w ramach systemu *Icinga*, jak i dodatku inGraph. Wyniki przetwarzania zostaną zapisane w odpowiednich bazach danych. Oznacza to, że całe przetwarzanie wykonywane jest na serwerze. Pozwala to na usunięcie funkcjonalności programu ingraph-collector z klienta mobilnego. Ponadto kolejną istotną różnicą jest możliwość umieszczenia bazy danych w sieci prywatnej. Jedynym elementem, który musi być dostępny z zewnątrz jest serwer monitorujący klientów mobilnych, gdyż do niego przesyłane są wyniki sprawdzeń. Warto również wspomnieć, że ilość przesyłanych danych została również ograniczona. W poprzedniej architekturze konieczne było przesłanie zarówno danych o stanie infrastruktury jak i danych potrzebnych do generacji wykresów. W konfiguracji z instancją nadrzędną do serwera przesyłane są jedynie rezultaty pomiarów, natomiast wszystkie pozostałe dane są na ich podstawie generowane przez serwer.

Idea działania tej architektury wydaje się być odpowiednia dla systemu monitorowania klienta mobilnego. Należy jednak dokonać analizy kluczowego dla tej konfiguracji dodatku NSCA. Jest on powszechnie wykorzystywany podczas monitorowania klientów statycznych, jednak specyficzne wymagania zdefiniowane dla klienta mobilnego mogą nie pozwolić na jego zastosowanie w tym systemie.

Rysunek 5.2. Monitoring klienta mobilnego w konfiguracji z instancją nadrzędną.



Po analizie przedstawionych w rozdziale 3 wymagań można stwierdzić, że dodatek NSCA nie spełnia bardzo wielu z nich. Głównymi problemami, które dyskryminują ten dodatek w zastosowaniu do monitorowania klienta mobilnego są:

Bezpieczeństwo. Mechanizmy bezpieczeństwa zawarte w protokole wymiany danych, przy zastosowaniu ich do komunikacji z klientem mobilnym posiadają poważne luki. Konieczność przechowywania na urządzeniu klucza symetrycznego, którego ujawnienie kompromituje cały system uniemożliwia realizację wymagania, które nie pozwala na kompromitację systemu po zgubieniu urządzenia.

Nadpisywanie stempla czasu. Moduł odbierający (NSCA) dodaje do każdego wpisu dziennika aktualny stempel czasu. Uniemożliwia to przesyłanie historycznych danych zgromadzonych wskutek utraty dostępu do sieci.

Brak dodatkowych mechanizmów uwierzytelnienia klienta. Decyzja o przydzieleniu klientowi dostępu, czyli akceptacji przesłanych przez niego danych, podejmowana jest na podstawie znajomości przez niego algorytmu szyfrowania oraz klucza symetrycznego.

Brak kontroli otrzymywanych danych. Każdy klient, który zna klucz może przesyłać dane dotyczące dowolnego urządzenia i dowolnej usługi. Brak mechanizmu, który pozwoliłby na kontrolę tego, jaki klient ma prawo informować o jakim urządzeniu czy też usłudze.

Brak potwierdzenia dostarczenia danych. Klient wysyłający dane nie ma żadnej informacji o tym, czy jego dane zostały zaakceptowane czy odrzucone. Oznacza to brak możliwości synchronizacji danych na kliencie mobilnym i serwerze, gdyż nigdy nie ma gwarancji, że wysłane przez klienta dane zostały przetworzone przez dodatek NSCA i przekazane do rdzenia monitorującego.

Brak implementacji dla systemów mobilnych. Moduł wysyłający jest aktualnie zaimplementowany jedynie na systemy Windows oraz Linux. Wiele współczesnych urządzeń mobilnych, które powinny być monitorowane, funkcjonuje pod kontrolą systemu operacyjnego Android czy też Windows Phone.

Przekazywanie danych tylko w jedno miejsce. Dane odebrane przez moduł odbierający mogą być przekazane w jedno miejsce. Przy bardziej złożonych systemach konieczne jest przekazywanie danych do kilku systemów oraz definiowanie reguł określających, które dane gdzie powinny trafić. Jest to przydatne na przykład do tworzenia kopii zapasowej danych otrzymanych od klienta.

Zastosowanie konfiguracji z nadrzędną instancją rdzenia monitorującego stanowi dobry szkielet dla systemu monitorowania klienta mobilnego. Niestety dostępne na rynku narzędzie, to jest dodatek NSCA, nie jest odpowiednio przystosowane do użycia ich w takim systemie. Wobec braku dostępnych narzędzi na rynku konieczne jest zaprojektowanie oraz zaimplementowanie nowego narzędzia, które spełni stawiane przed nim wymagania. W ramach tej pracy wykonano projekt oraz implementację dodatku NSCAv2, który uwzględnia dodatkowe wymagania wynikające z monitorowania klienta mobilnego. Ponieważ protokół komunikacyjny używany w dodatku NSCA jest również niedostosowany do potrzeb klienta mobilnego, w ramach tej pracy zaproponowano nowy protokół komunikacyjny pozwalający na niezawodną transmisję danych z urządzenia mobilnego do serwera.

Należy również zaznaczyć, że w chwili pisania tej pracy, nie udało się odnaleźć na rynku żadnej aplikacji pozwalającej na monitorowanie urządzenia mobilnego i współpracującej w jakikolwiek sposób z systemem *Icinga*. Ze względu na brak takiej aplikacji w niniejszej pracy dokonano zarysu wymaganej od takiej aplikacji funkcjonalności. Aplikacja realizująca i rozszerzająca tą funkcjonalność została zaprojektowana i zaimplementowana przez Pana Marcina Kubika w ramach pracy inżynierskiej [22] realizowanej na Wydziale Elektroniki i Technik Informacyjnych.

5.2. Protokół komunikacyjny

Wymagania przedstawione w rozdz. 3 definiują bardzo wiele cech systemu, które muszą być zapewnione poprzez użycie odpowiedniego protokołu komunikacyjnego. Analiza wymagań pozwoliła na wyodrębnienie następujących cech protokołu komunikacyjnego:

spójność danych — protokół musi gwarantować, że dane zostały dostarczone i przetworzone;

integralność — protokół musi zapewniać, że dane zostaną dostarczone w niezmodyfikowanej postaci i jedynie od uwierzytelnionego nadawcy;

poufność protokół musi zapewniać przekazanie danych w sposób, który uniemożliwi stroną trzecim ich odczytanie;
niezależność algorytmu szyfrowania — protokół musi być niezależny od algorytmu, którym szyfrowane są dane;
uwierzytelnienie klienta — protokół musi zapewniać element pozwalający na potwierdzenie tożsamości klienta;
niezależność uwierzytelnienia klienta — protokół musi być niezależny od wykorzystywanej metody uwierzytelnienia klienta;
uwierzytelnienie serwera — protokół musi zapewniać potwierdzenie tożsamość serwera;
odporność na utratę urządzenia klienckiego — protokół nie może wymagać przechowywania na urządzeniu danych pozwalających na kompromitację całego systemu;
oszczędność pasma — protokół powinien minimalizować ilość przesyłanych danych.

Rozbudowane wymagania bezpieczeństwa protokołu wynikają z charakterystyki przesyłanych danych. Dane, które pochodzą z urządzenia, mogą zawierać zarówno poufne dane właściciela, jak i tajemnice handlowe firmy. Ujawnienie tych danych może pociągać za sobą poważne konsekwencje finansowe lub prawne, dlatego konieczne jest zapewnienie bezpiecznego protokołu. Należy również pamiętać, że jedna ze stron komunikujących się przy użyciu protokołu znajduje się na urządzeniu mobilnym, dlatego należy ograniczyć narzut wprowadzany przez użycie tego protokołu.

Spośród bezpiecznych protokołów komunikacyjnych rozpowszechnionych na rynku najbliższym spełnienia wszystkich wymagań jest protokół Secure Socket Layer - SSL[1]. Jest to protokół warstwy prezentacji, który pozwala na bezpieczny transport strumienia danych. Protokół wykorzystuje kryptografię symetryczną i asymetryczną. Kryptografia asymetryczna wykorzystywana jest do uwierzytelnienia serwera i opcjonalnie klienta przy pomocy certyfikatów nadawanych przez centra certyfikacji. Model bezpieczeństwa zastosowany w tym protokole pozwala przy pomocy kluczy centrów autoryzacji dokonywać weryfikacji certyfikatów przesyłanych przez wiele witryn. Niestety głębsza analiza protokołu wykazała, iż nie spełnia on wszystkich wymagań. Przede wszystkim zestawienie połączenia wymaga przesłania znaczącej ilości danych. Ponadto konieczne jest zdobycie certyfikatu, który pozwalałby na weryfikację serwera. Model bezpieczeństwa zastosowany w SSL jest dla rozpatrywanego przypadku nadmiarowy, ponieważ klient mobilny przekazuje dane zawsze do tego samego serwera. A to z kolei powoduje nadmierne zużycie pasma. Ponadto zamknięty zbiór algorytmów szyfrowania możliwych do wykorzystania powoduje, że nie może on być zastosowany w omawianym przypadku. Wobec braku gotowego protokołu konieczne jest zaprojektowanie nowego, który spełni wszystkie stawiane wymagania.

Protokół taki, został opracowany w ramach niniejszej pracy. Jest on oparty na protokole TCP, który zapewnia abstrakcję przesłania strumienia bajtów z gwarancją ich dostarczenia. Mnogość wymagań dotyczących projektowanego protokołu utrudnia wykorzystanie architektury jednowarstwowej. Konieczne jest zatem wydzielenie warstw, z których każda będzie zapewniała dobrze zdefiniowane usługi dla warstw wyższych.

[1] Szczegółowy opis protokołu można znaleźć w [21, 148-155].

5.2.1. Warstwa formowania wiadomości

Najniższa warstwa protokołu komunikacyjnego zbudowana jest bezpośrednio na protokole TCP. Komunikujące się strony w swej architekturze wykorzystują paradygmat programowania zdarzeniowego. Abstrakcja strumienia bajtów zapewniana przez protokół TCP nie jest odpowiednia dla tego modelu. Konieczne jest zatem dostarczenie warstwy, która umożliwi przesłanie w całości komunikatu o zadanej długości. Umożliwia to wygodne przesyłanie wiadomości odpowiadających poszczególnym zdarzeniom w komunikujących się programach.

Usługa zapewniana przez tą warstwę jest bardzo prosta, dzięki czemu rozpoczęcie komunikacji nie wymaga żadnej inicjalizacji. Protokół jest w pełni symetryczny. Oznacza to, że obie komunikujące się strony posiadają taki sam dozwolony zbiór stanów protokołu. Warstwa świadczy usługę przekazywania wiadomości o zdefiniowanym rozmiarze. W celu wykonania tej usługi, do danych, które są dostarczone stronie nadawczej, dołączana jest ich długość. Długość w protokole jest reprezentowana jako 32-bitowa liczba ze znakiem o sieciowej kolejności bajtów. Tak sformatowana wiadomość przesyłana jest przy użyciu protokołu TCP do odbiorcy. Odbiorca po odebraniu pierwszych czterech bajtów wiadomości sprawdza rozmiar danych, po czym rozpoczyna odbieranie ilości danych określonej przez nagłówek. Wiadomość jest przekazywana użytkownikowi dopiero w momencie odebrania całego komunikatu. Jeśli w trakcie odbierania fragmentów wiadomości nastąpi przerwanie połączenia, odebrane fragmenty wiadomości są porzucane, a użytkownikowi sygnalizowany jest błąd.

Długość danych	Dane

Tablica 5.1: Struktura komunikatu warstwy formowania wiadomości.

5.2.2. Warstwa kryptograficzna

Warstwa ta jest odpowiedzialna za zapewnienie poufności oraz integralności przesyłanych danych. Ponadto zadaniem tej warstwy jest również wykonanie uwierzytelnienia serwera. Podczas projektowania tej warstwy konieczne było również uwzględnienie wymagania, które zalecało niezależność algorytmu szyfrowania przesyłanych danych od protokołu komunikacyjnego. W celu zapewnienia możliwości późniejszej modyfikacji protokołu, warstwa ta zawiera również proces negocjacji wersji protokołu.

Model bezpieczeństwa implementowany przez tę warstwę jest zbliżony do protokołu SSL, jednak zostały wprowadzone zmiany, które zmniejszają zużycie pasma oraz eliminują potrzebę wykorzystania certyfikatów. Zanim możliwa będzie bezpieczna komunikacja z użyciem tej warstwy, konieczne jest umieszczenie na urządzeniu mobilnym klucza publicznego RSA serwera oraz przechowywanie klucza prywatnego na serwerze. Istotne jest, aby klucze mogły być umieszczane jedynie przez autoryzowaną osobę, np. administratora tych urządzeń. Bezpośrednie umieszczenie klucza publicznego serwera na urządzeniu eliminuje potrzebę wykorzystania certyfikatów oraz ich przesyłania. Należy jednak zwrócić uwagę, iż w przyjętym modelu nie jest możliwa zmiana klucza publicznego serwera bez ponownego umieszczenia go na urządzeniu. Nie stanowi to jednak problemu, gdyż zmiana klucza publicznego w projektowanym systemie jest sytuacją niezwykle rzadką. Ponieważ szyfrowanie asymetryczne wymaga znacznie większego narzutu obliczeniowego

jest ono używane tylko w trakcie nawiązywania połączenia. Właściwy transport danych szyfrowany jest przy pomocy uzgodnionego klucza symetrycznego. W poniższym omówieniu protokołu, a także na diagramach, w celu uproszczenia, pominięto fakt istnienia limitu czasu oczekiwania oraz możliwość rozłączenia w dowolnym momencie. Obie te sytuacje powodują zakończenie działania i zgłoszenie błędu warstwie wyższej. Uproszczona wersja maszyny stanowej procesu inicjalizacji komunikacji w tej warstwie po stronie serwera znajduje się na rys. 5.3, a po stronie klienta na rys. 5.4.

Komunikacja z użyciem tej warstwy możliwa jest dopiero po wykonaniu inicjalizacji. Proces inicjalizacji rozpoczyna się od negocjacji używanej wersji protokołu. Niezwłocznie po nadejściu połączenia od klienta serwer wysyła komunikat będący zapytaniem o żądaną przez klienta wersję protokołu. Klient odpowiada na ten komunikat, przesyłając żądaną wersję protokołu. Jeśli serwer może obsłużyć daną wersję protokołu, przesyła on pozytywne potwierdzenie do klienta, co powoduje rozpoczęcie kolejnego etapu inicjalizacji. W przeciwnym przypadku serwer przesyła informację o odrzuceniu żądania. Po odebraniu negatywnego potwierdzenia klient może podjąć kolejne próby, używając innych wersji protokołu. Dalsza komunikacja uzależniona jest od wybranej wersji protokołu komunikacyjnego.

Kod REQUEST_PROTOCOL	Nazwa wersji

Tablica 5.2: Struktura komunikatu żądanie wersji.

W zaproponowanej w tej pracy wersji protokołu kolejnym etapem inicjalizacji połączenia jest uwierzytelnienie serwera połączone z przedstawieniem klienta. Etap ten rozpoczyna się od przesłania przez klienta jego identyfikatora oraz losowych ośmiu bajtów danych. Komunikat zaszyfrowany jest kluczem publicznym serwera, zatem może go odczytać jedynie posiadacz odpowiedniego klucza prywatnego. Serwer po odebraniu wiadomości odszyfrowuje ją, używając swojego klucza prywatnego. Jeśli wiadomość jest nieczytelna lub klient o podanym identyfikatorze nie istnieje, serwer niezwłocznie kończy połączenie. Jeśli wiadomość jest poprawna, a klient o podanym identyfikatorze istnieje, serwer wykonuje podpis cyfrowy identyfikatora oraz losowych bajtów odebranych od klienta. Do klienta odsyłany jest komunikat zawierający wykonany przez serwer podpis. Klient po odebraniu podpisu wykonuje weryfikację podpisu na podstawie wiadomości, która została przesłana do serwera. Jeśli podpis jest zgodny, oznacza to, że urządzenie, z którym nastąpiło połączenie, jest posiadaczem odpowiedniego klucza prywatnego, czyli zostało upoważnione przez administratora do odbierania danych pochodzących od tego klienta.

Kod CLIENT_ID	Ciąg losowy	Identyfikator

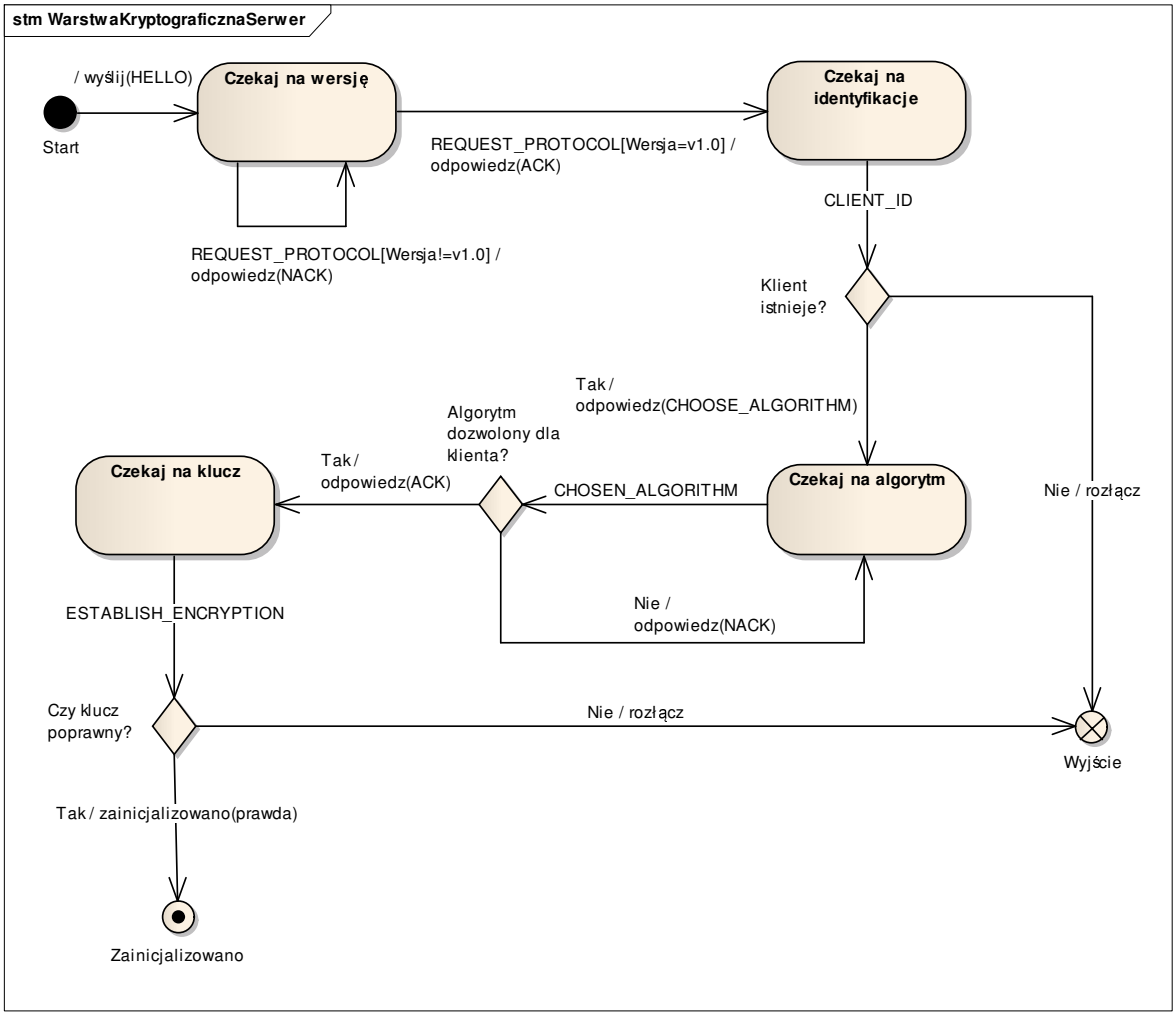
Tablica 5.3: Struktura komunikatu identyfikatora klienta.

Kod CHOOSE_ALGORITHM	Podpis danych z CLIENT_ID

Tablica 5.4: Struktura komunikatu potwierdzającego identyfikator.

Ostatnim etapem inicjalizacji komunikacji w tej warstwie jest negocjacja algorytmu szyfrowania oraz generacja odpowiedniego klucza symetrycznego. Klient

Rysunek 5.3. Maszyna stanów warstwy kryptograficznej po stronie serwera.

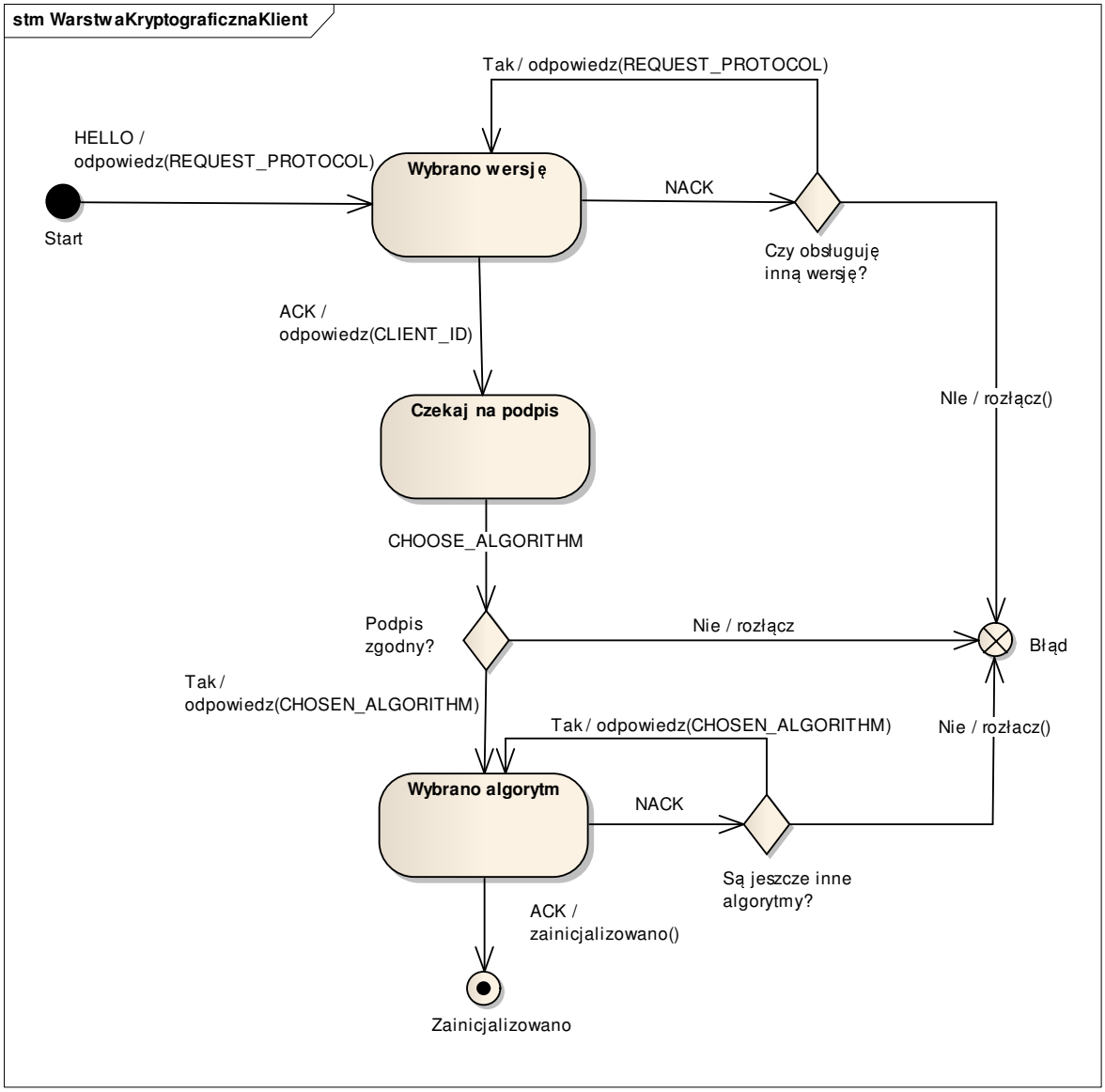


przesyła do serwera zaszyfrowany kluczem publicznym komunikat zawierający żądanie algorytmu symetrycznego. Serwer po odebraniu komunikatu odszyfrowuje go przy użyciu klucza prywatnego. W zależności od dostępności żądanego algorytmu, do klienta odsyłany jest komunikat akceptujący lub odrzucający wybrany algorytm. Klient po odebraniu negatywnego potwierdzenia może ponownie zażądać innego algorytmu symetrycznego. Klient po odebraniu komunikatu akceptującego wybrany algorytm dokonuje generacji klucza symetrycznego oraz wektora wartości początkowych jeśli jest to wymagane przez tryb pracy algorytmu. Następnie obliczany jest skrót całej wiadomości. Tak przygotowany komunikat szyfrowany jest kluczem publicznym i przesyłany do serwera. Serwer po odebraniu komunikatu sprawdza poprawność klucza oraz wektora początkowego oraz zgodność danych z dołączonym skrótem, co stanowi ostatni etap inicjalizacji.

Kod CHOSEN_ALGORITHM	Nazwa algorytmu

Tablica 5.5: Struktura komunikatu żądania algorytmu.

Rysunek 5.4. Maszyna stanów warstwy kryptograficznej po stronie klienta.



Długość skrótu	Kod ESTABLISH_ENCRYPTION	Długość klucza	Klucz symetryczny	Wektor początkowy	Skrót

Tablica 5.6: Struktura komunikatu zawierającego klucz symetryczny.

Wykonanie inicjalizacji pozwoliło na uzgodnienie w sposób bezpieczny klucza symetrycznego. W dalszej komunikacji wszystkie komunikaty szyfrowane są z użyciem wybranego algorytmu symetrycznego. Ponieważ algorytmy szyfrowania nie zapewniają integralności przesyłanych danych, konieczne jest użycie funkcji skrótu. W omawianym protokole została użyta funkcja SHA2. Przygotowanie zatem każdego komunikatu z danymi rozpoczyna się od obliczenia skrótu danych. Ponieważ algorytm skrótu nie był negocjowany, konieczne jest dostarczenie długości używanego skrótu. Długość skrótu wyrażona w bajtach dołączana jest na początku wiadomości, natomiast sam skrót na końcu. Komunikat przed wysłaniem szyfrowany jest uzgodnionym kluczem symetrycznym.

Długość skrótu	Dane	Skrót danych

Tablica 5.7: Struktura komunikatu danych warstwy kryptograficznej.

5.2.3. Warstwa transportu pomiarów

Warstwa ta zapewnia zapewnia transport danych o stanie urządzeń i usług w pakietach o dowolnym rozmiarze. Wykorzystanie warstw niższych gwarantuje zarówno poufność i integralność przesyłanych komunikatów. Żadna z niższych warstw nie zapewnia jednak uwierzytelnienia klienta, dlatego jest to również jedno z zadań tej warstwy.

Inicjalizacja komunikacji w tej warstwie rozpoczyna się od negocjacji algorytmu uwierzytelnienia klienta. Serwer przesyła do klienta komunikat informujący o konieczności wyboru algorytmu uwierzytelnienia. Klient przesyła komunikat zawierający nazwę algorytmu, który ma być użyty do potwierdzenia tożsamości. Serwer po otrzymaniu komunikatu sprawdza, czy żądany algorytm jest dostępny dla tego klienta. Jeśli nie, przesyłany jest komunikat informujący o odrzuceniu żądania, a klient może ponowić żądanie, używając innego algorytmu. Jeśli algorytm wybrany przez klienta jest dostępny, rozpoczyna się proces uwierzytelnienia.

Kod CHOSEN_AUTH_MODULE	Nazwa algorytmu uwierzytelnienia

Tablica 5.8: Struktura komunikatu żądania algorytmu uwierzytelnienia.

Uwierzytelnienie klienta wykonywane jest poprzez zewnętrzne moduły, gdyż protokół komunikacyjny musi być niezależny od protokołu komunikacyjnego. Algorytm uwierzytelnienia uprawniony jest do przesyłania dowolnych danych w obie strony. Jeśli algorytm uwierzytelnienia odrzuci klienta, oznacza to natychmiastowe zamknięcie połączenia. Pomyślne zakończenie procesu uwierzytelnienia oznacza konieczność wykonania sprawdzenia, czy klient posiada zdefiniowane miejsca, do których może przekazywać swoje dane. W przypadku braku takiego miejsca, aby dane nie zostały utracone, do klienta wysyłany jest komunikat negatywnego potwierdzenia, a połączenie jest zamykane. Jeśli co najmniej jedno miejsce docelowe zostało zdefiniowane, do klienta wysyłany jest komunikat informujący o oczekiwaniu na przesłanie danych, co kończy proces inicjalizacji. Przebieg omówionego procesu inicjalizacji oraz pozostałych elementów protokołu po stronie serwera przedstawia rys. 5.5, natomiast po stronie klient rys. 5.6.

Kod AUTH_DATA	Dane

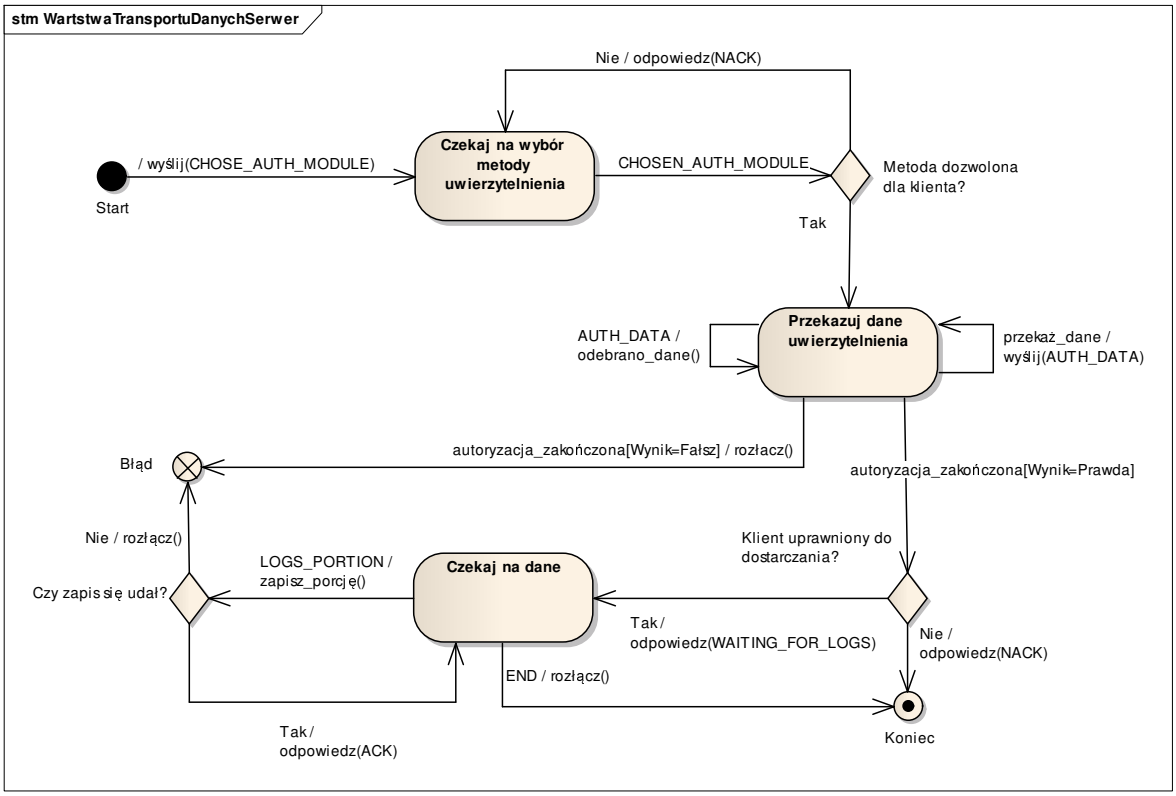
Tablica 5.9: Struktura komunikatu z danymi uwierzytelnienia.

Dane przesyłane przez klienta znajdują się w pakietach. Po odebraniu każdego pakietu i jego pomyślnym przetworzeniu przez aplikację obliczany jest skrót danych w nim zawartych. Obliczony skrót jest następnie przesyłany w pakiecie potwierdzającym przetworzenie danych.

Kod LOGS_PORTION	Dane pomiarów

Tablica 5.10: Struktura komunikatu zawierającego dane.

Rysunek 5.5. Maszyna stanów warstwy transportu pomiarów po stronie serwera.



Pakiet z danymi może zawierać dowolną ilość danych. Każdy wynik sprawdzenia powinien posiadać odpowiedni format. W szczególności powinien zawierać bajt danych, który pozwala na określenie, czy dotyczy on urządzenia, czy usługi. Niezbędne jest również przesłanie stempla czasu, który determinuje, kiedy dane sprawdzenie zostało wykonane. Stempel ten powinien być zgodny ze stemplem czasu systemu Unix, a jego przesłanie musi odbywać się w sieciowej kolejności bajtów. Do prawidłowego przekazania danych do systemu monitorującego konieczne jest również dostarczenie nazwy urządzenia oraz usługi. Aby oddzielić te dane konieczne jest zakończenie każdego z tych pól znakiem zerowym. Każdy wpis powinien również zawierać bajt informujący o stanie, w jakim jest dana usługa. Kod ten powinien być zapisany przy użyciu jednego bajtu w sposób zgodny z kodami akceptowanymi przez system monitorujący.

Kod HOST	Stempel czasu	Nazwa urządzenia	Kod stanu	Wpis

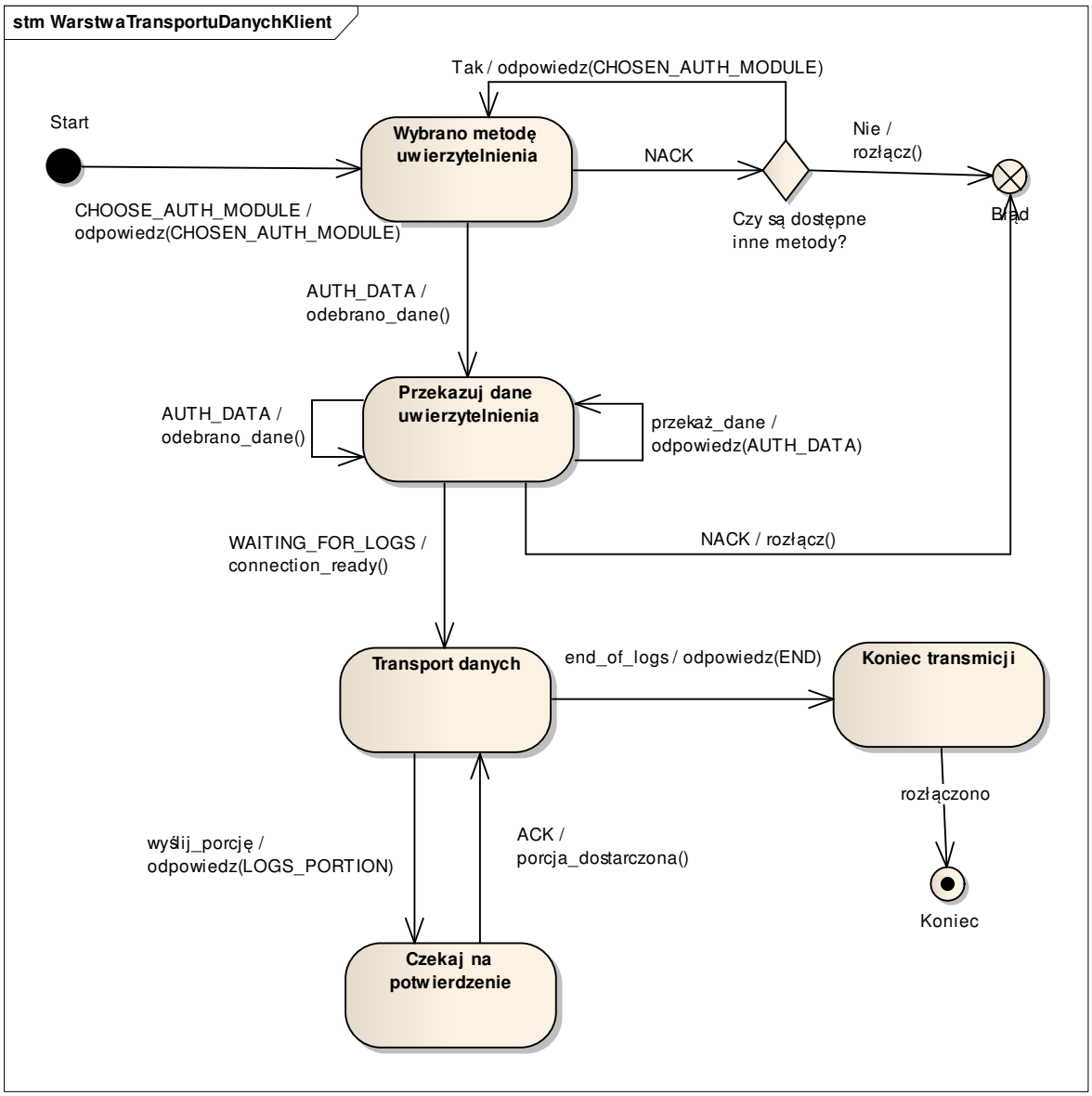
Tablica 5.11: Struktura pojedynczego wpisu dziennika urządzenia.

Kod SERVICE	Stempel czasu	Nazwa urządzenia	Nazwa usługi	Kod stanu	Wpis

Tablica 5.12: Struktura pojedynczego wpisu dziennika usługi.

Jeśli odebrane dane posiadają nieprawidłową strukturę lub klient dokonał próby przesłania danych, do których przesyłania nie ma uprawnień, połączenie jest natychmiast zamykane. Klient po przesłaniu wszystkich danych lub w chwili

Rysunek 5.6. Maszyna stanów warstwy transportu pomiarów po stronie klienta.



zdefiniowanej przez administratora może zdecydować o zamknięciu połączenia, wysyłając odpowiedni komunikat. Serwer po odebraniu tego komunikatu zamknie połączenie.

5.3. Zarys aplikacji mobilnej IcingaMini

Aplikacja IcingaMini jest odpowiedzialna za monitorowanie zadanych parametrów urządzenia mobilnego. Klienty mobilne są wzajemnie niezależne i mogą operować bez możliwości komunikacji pomiędzy sobą. Konieczne jest zatem, aby każdy klient mobilny posiadał swoją instancję tej aplikacji. Możemy wyróżnić trzy podstawowe, rozdzielne funkcjonalnie elementy:

— element pomiarowy,
— element komunikacyjny,

— zestaw wtyczek.

Element pomiarowy jest odpowiedzialny za planowanie i wykonywanie pomiarów zgodnie z polityką zdefiniowaną przed administratora. Ponadto konieczne jest zapewnienie składowania uzyskanych informacji do czasu udanej synchronizacji z serwerem. Element komunikacyjny jest to implementacja opisanego wcześniej protokołu komunikacyjnego dla danej platformy. Zadaniem tej części jest dostarczenie wyników pomiarów do miejsc zdefiniowanych przez administratora. Wtyczki są to elementy bezpośrednio odpowiedzialne za wykonywanie pomiarów zadanych wartości czy testowanie zdefiniowanych usług. Metoda realizacji wtyczek uzależniona jest od platformy sprzętowej na jaką przeznaczona jest dana aplikacja. Konieczne jest jednak zapewnienie niezależności elementu pomiarowego od zestawu wykorzystywanych wtyczek, aby umożliwić swobodną zmianę zbioru wykorzystywanych wtyczek.

Implementacja tego modułu musi uwzględniać uwarunkowania sprzętowe jak i systemowe platformy, na której się znajduje. Urządzenia mobilne są zazwyczaj zasilane z własnych akumulatorów dlatego konieczne jest zastosowanie mechanizmów, które pozwolą na zredukowanie zużycia energii związanego z systematycznym wykonywaniem sprawdzeń. Należy również wspomnieć, iż moduł mobilny odpowiedzialny jest za nadawanie każdemu z odczytów stempla czasu uniwersalnego[2] dokonywanego pomiaru. Na podstawie dokonanej charakterystyki klienta mobilnego, można poczynić założenie, iż posiada on dostęp do punktów synchronizacji czasu. Jest wiele dostępnych metod synchronizacji czasu na urządzeniu mobilnym, między innymi pobranie czasu z sieci GSM czy też z serwerów czasu światowego, przez co nie stanowi to dla aplikacji poważnego wymagania.

Aplikacja IcingaMini po zebraniu porcji wpisów dziennika o rozmiarze zgodnym z polityką administratora, lub po upływie określonego czasu powinna przesłać posiadane wpisy dziennika do modułu odbiorczego, a po uzyskaniu potwierdzenia usunąć je z urządzenia w celu oszczędności pamięci. Możliwa jest również sytuacja, w której urządzenie mobilne użytkowany jest przez pewien czas bez dostępu do sieci, przez którą możliwa jest komunikacja z serwerem. W takiej sytuacji aplikacja powinna gromadzić odczyty, aż do czasu uzyskania możliwości połączenia z serwerem. Różnorodność platform dostępnych na rynku sprawia, iż nie można wymagać dostarczenia uniwersalnej implementacji protokołu komunikacyjnego. Wymaga się zatem, aby każda implementacja aplikacji mobilnej używała protokołu komunikacyjnego zgodnego z protokołem dodatku NSCAv2 opisanego w rozdziale 5.2. Należy również zapewnić możliwość weryfikacji tożsamości serwera, z którym nawiązuje się połączenie.

W ramach systemu monitorowania możliwe jest funkcjonowanie wielu urządzeń wykorzystujących tą aplikację. Urządzenia te mogą wykorzystywać bardzo wiele różnych platform (Android, Windows Phone, Ubuntu). W chwili pisania tej pracy nie znaleziono na rynku żadnej aplikacji przeznaczonej na platformę mobilną, która spełniałaby stawiane wymagania. Szczegółowy projekt oraz implementacja aplikacji spełniającej przedstawione tu wymagania wykracza poza zakres niniejszej pracy. Podczas okresu testowania wykonanego systemu wykorzystano aplikację, przeznaczoną dla platformy Android. Została ona zaprojektowany i zaimplementowana przez Pana Marcina Kubika. Szczegółowy opis jej implementacji można znaleźć w [22].

[2] Czas uniwersalny - średni astronomiczny czas słoneczny na południku zerowym.

5.4. Koncepcja dodatku NSCAv2

Dodatek NSCAv2 jest to zaimplementowany w ramach tej pracy zamiennik dla dodatku NSCA. Podobnie jak NSCA dodatek ten służy do przekazywania wyników pasywnego monitorowania z urządzeń innych, niż to na którym uruchomiony jest system monitorujący. Istotną różnicą pomiędzy tymi dodatkami jest wykorzystywany protokół komunikacyjny. W dodatku NSCAv2 zaimplementowano opracowany w ramach tej pracy protokół komunikacyjny (patrz rozdział 5.2).

NSCAv2 w znaczący sposób różni się pod względem koncepcyjnym od swojego poprzednika. Przede wszystkim dodano logiczną reprezentację klienta[3] jako posiadacza kilku urządzeń. Oznacza to zdefiniowanie następującej hierarchii bytów w programie:

klient — posiadacz pewnej liczby urządzeń, którego tożsamość jest potwierdzana przez proces uwierzytelnienia.

urządzenie — urządzenie mobilne lub stacjonarne o nazwie zdefiniowanej w konfiguracji programu monitorującego.

usługa — usługa sieciowa lub parametr urządzenia monitorowany przez system.

Każdy klient może posiadać wiele urządzeń, a każde urządzenie wiele usług. Klient może przekazywać dane dotyczące pomiarów jedynie dla urządzeń i usług, które zostały do tego klienta przypisane przez administratora w konfiguracji dodatku. Wszystkie dane przesyłane przez klienta są sprawdzane i jeśli nastąpi próba przekazania danych dotyczących usługi lub urządzenia, do którego klient nie ma uprawnień, dane te zostaną odrzucone.

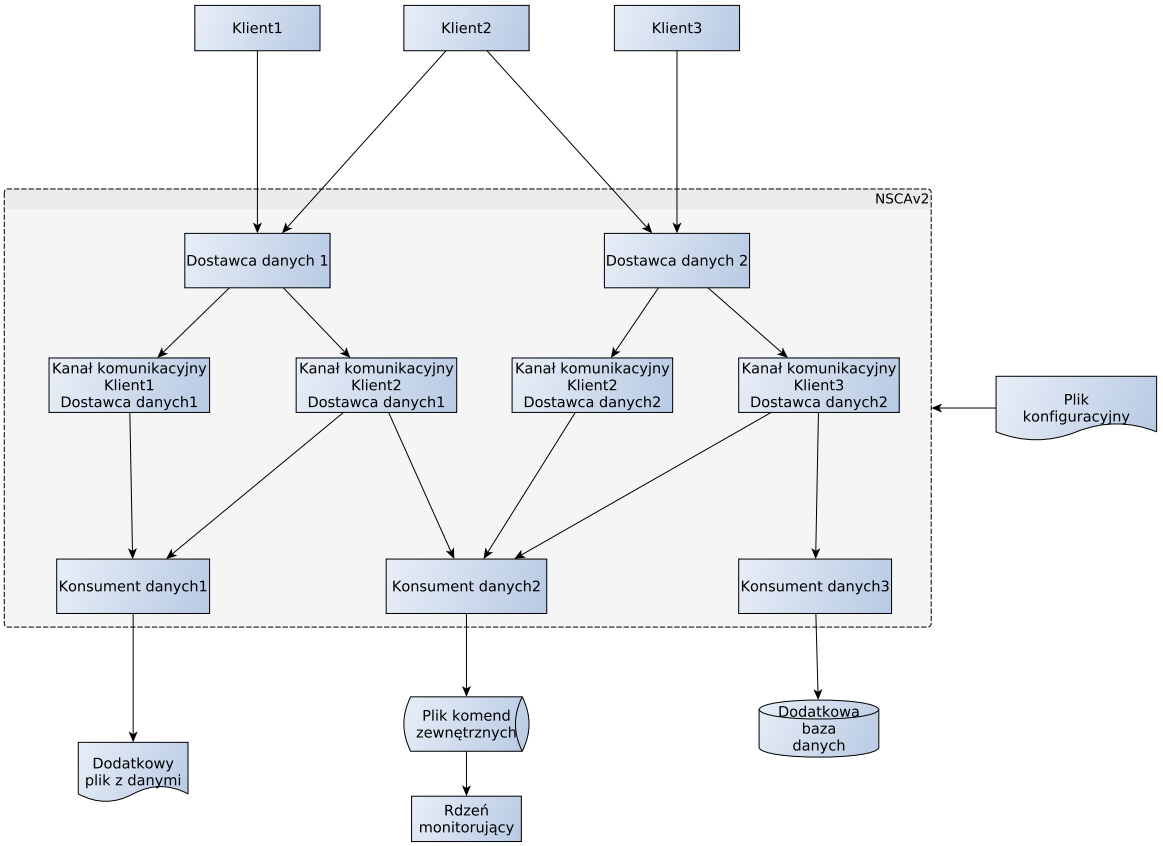
Dodatek NSCAv2 uwzględnia wszystkie wymagania stawiane przed systemem monitorowania klienta mobilnego. Jest to zauważalne w jego architekturze logicznej. Dodatek NSCA posiadał budowę monolityczną, przez co możliwe było dostarczanie do niego danych tylko i wyłącznie jedną drogą (zdefiniowany przez dodatek protokół komunikacyjny) i przekazywanie ich jedynie do jednego miejsca (plik komend zewnętrznych — patrz 4). NSCAv2 posiada natomiast budowę modularną, której schemat logiczny został przedstawiony na rys. 5.7.

Elementem dodatku NSCAv2, który odbiera dane od klienta jest dostawca danych, który moze implementować dowolny protokół komunikacyjny. W ramach tej pracy zaimplementowano protokół opisany w rozdz. 5.2. Architektura NSCAv2 zakłada jednak możliwość istnienia wielu niezależnych od siebie instancji dostawców danych (implementujących różne protokoły transmisji lub nasłuchujących na różych portach).

Modułem odpowiedzialnym, za przekazywanie danych do systemu monitorującego jest konsument danych. Jest to implementacja metody przekazywania danych w zależności od miejsca docelowego. W ramach tej pracy dostarczono konsumenta danych, który zapisuje otrzymane dane do pliku komend zewnętrznych systemu *Icinga* oraz konsumenta danych, który wypisuje otrzymane dane w postaci logów programu. Konsumentów danych również może być więcej niż jeden.

[3] Należy zwrócić uwagę na różnicę znaczeniową pomiędzy klient — osoba posiadająca urządzenie oraz klient mobilny czyli urządzenie mobilne, a także klient statyczny czyli urządzenie statyczne, stacjonarne (patrz rozdział 3).

Rysunek 5.7. Diagram architektury modułu odbiorczego



Istnienie wielu konsumentów oraz wielu dostawców danych powoduje konieczność dodania obiektu, który będzie definiował powiązania pomiędzy tymi elementami. Elementem tym jest kanał komunikacyjny. Jest to obiekt, który na podstawie informacji o dostawcy oraz kliencie determinuje do jakich konsumentów mają zostać przekazane dane. Reguły te definiowane są przez administratora w pliku konfiguracyjnym.

Wymagania przedstawione w rozdzale 3 wymuszają umożliwienie dodawania nowych algorytmów zarówno kryptograficznych, jak i algorytmów uwierzytelnienie klienta. Aby spełnić wymagania, wyróżnione zostały w programie dwa moduły pomocnicze:

— moduł uwierzytelnienia,
— moduł kryptograficzny.

Moduł uwierzytelnienia stanowi bibliotekę algorytmów uwierzytelnienia klienta, które mogą być wykorzystane przez dostawców danych w celu weryfikacji tożsamości klienta. Moduł kryptograficzny stanowi natomiast bibliotekę algorytmów kryptografii symetrycznej, asymetrycznej oraz funkcji skrótu. Umożliwia to realizację procesu negocjacji algorytmu symetrycznego wykorzystywanego przez protokół komunikacyjny. Ponieważ różnice pomiędzy poszczególnymi protokołami komunikacyjnymi mogą być niewielkie, w programie wyodrębniono również moduł implementujący poszczególne warstwy protokołu. Umożliwi to w przyszłości szybką modyfikację protokołu komunikacyjnego lub dodanie nowego.

6. Implementacja

W rozdziale 5 przedstawiono autorskie rozwiązanie problemu monitorowania klienta mobilnego przy użyciu systemu *Icinga*. W ramach projektowania takiego systemu dokonano analizy dostępnych na rynku dodatków pod kątem ich przydatności w systemie monitorowania klienta mobilnego. Przeprowadzona analiza wykazała konieczność opracowania nowego dodatku, pozwalającego na przekazywanie w sposób zgodny z wymaganiami, danych z urządzenia mobilnego do serwera. Dodatek ten — NSCAv2 został zaprojektowany i zaimplementowany w ramach niniejszej pracy. Rozdział ten zawiera dokładny opis funkcjonalny tego dodatku oraz przedstawia istotne elementy jego implementacji.

Ponadto, w proponowanym systemie występuje aplikacja przeznaczona dla platformy mobilnej. Implementacja takiej aplikacji wykracza poza zakres tej pracy. W ramach serii prac dyplomowych przygotowywanych na Wydziale Elektroniki i Technik Informacyjnych powstała również praca inżynierska Pana Marcina Kubika. Przedmiotem tej pracy jest projekt oraz implementacja dla platformy Android aplikacji IcingaMini. Pobieżny opis koncepcji i potrzeby tej aplikacji można znaleźć w rozdz. 5.3 natomiast szczegółowy opis implementacji został zawarty w [22].

6.1. Opis architektury

Konceptualny model dodatku został przedstawiony w rozdz. 5.4. Fizyczna struktura programu jest dużo bardziej złożona. Została utworzona z użyciem biblioteki Qt jako podstawowego szkieletu aplikacji. Wykorzystano również biblioteki boost oraz Crypto++. Dodatek ten przeznaczony jest, podobnie jak system monitorujący *Icinga*, dla komputerów pracujących pod kontrolą systemu operacyjnego Linux i jest uruchamiany jako samodzielny serwis. Fizyczna struktura programu składa się z następujących elementów:

szkielet programu — Zawiera elementy programu konieczne do wytworzenia środowiska dla funkcjonowania pozostałych modułów oraz zarządzania nimi. Ponadto zawiera odwzorowanie bytów logicznych (dostawca, konsument, kanał komunikacyjny) programu w strukturę fizyczną.

moduł kryptograficzny — Dostarcza implementacji funkcji kryptograficznych, wymaganych podczas komunikacji z klientem mobilnym. Zawiera zarówno algorytmy asymetryczne, konieczne do inicjalizacji kryptografii symetrycznej, jak i algorytmy symetryczne, służące do przesyłania danych.

moduł autoryzacji klienta — Zawiera implementację algorytmów uwierzytelnienia klienta.

moduł komunikacji z użyciem TCP — Dostarcza implementację protokołu komunikacyjnego używanego do komunikacji z klientem.

moduł logowania — Pozwala na przekazywanie użytkownikowi komunikatów z dowolnych miejsc znajdujących się w innych modułach. Wiadomość ta może zawierać informacje o zaistniałym błędzie lub innym zdarzeniu, wymagającym poinformowania użytkownika.

Odwzorowanie podstawowych elementów struktury logicznej w fizyczną ma miejsce w szkielecie aplikacji. Pozostałe elementy programu zostały zaprojektowane jako moduły pomocnicze świadczące dobrze zdefiniowane usługi. Szczególnym przykładem tego usługowego charakteru pozostałych modułów może być moduł kryptograficzny i moduł autoryzacji klienta. Udostępniają one generyczne interfejsy dostępu do swoich usług dla pozostałych modułów. Szczegóły implementacyjne są natomiast zamknięte wewnątrz modułów. Typ faktyczny obiektu udostępnianego poprzez generyczny interfejs jest w bardzo wielu przypadkach determinowany dopiero na etapie konfiguracji programu. Ponadto liczba klas znajdujących się w tych modułach może znacząco wzrastać. Jednym z wymagań było zapewnienie możliwości definiowania nowych algorytmów kryptograficznych jak i algorytmów uwierzytelnienia klienta. Aby można było wybrać typ faktyczny obiektu na podstawie danych, wykorzystano wzorzec projektowy fabryki[1]. Każdy ze wspomnianych modułów posiada swojego zarządcę (fabrykę). Dostępne w module obiekty muszą zostać zarejestrowane u swojego zarządcy. Pozostałe moduły uzyskują instancje tych obiektów poprzez zarządcę, który na podstawie przekazanych danych określa typ faktyczny obiektu. Jeśli dany typ jest dostępny, zostanie on wówczas przekazany wywołującemu i będzie on mógł używać go poprzez dostarczony generyczny interfejs. Logiczna struktura tych modułów wymaga zapewnienia, że w programie istnieje tylko jedna instancja obiektu danego zarządcy. W tym celu został wykorzystany wzorzec projektowy singleton[2]. Zastosowanie wzorca projektowego fabryki pozwala pozostałym modułom na korzystanie z obiektów, których typ faktyczny jest nieznany w trakcie implementacji oraz kompilacji programu.

W celu konfiguracji programu wykorzystano zewnętrzny plik w formacie XML. Umożliwia to zmianę ustawień programu bez konieczności jego ponownej kompilacji. Plik konfiguracyjny składa się z czterech zasadniczych sekcji:

sekcja dostawców danych — zawiera dane dostawców, którzy mają zostać uruchomieni podczas startu programu. Umożliwia przekazanie dodatkowych informacji do obiektu dostawcy, np. adresu IP lub portu, na którym powinien on oczekiwać na połączenia.

sekcja odbiorców danych — zawiera dane odbiorców danych, którzy mają zostać uruchomieni podczas startu programu. Umożliwia przekazanie dodatkowych informacji do obiektu odbiorcy danych, takich jak ścieżka do pliku, do którego należy zapisywać dane.

sekcja definicji klientów — zawiera definicję klientów oraz grup klientów. Każda definicja klienta składa się z następujących sekcji:

— sekcja autoryzacji zawiera dane o dozwolonych modułach autoryzacyjnych dla danego klienta. Umożliwia także dodatkową konfigurację instancji modułów przeznaczonych dla danego klienta.

— sekcja filtrowania zawiera urządzenia oraz usługi, których dane monitorowania mogą być przesyłane przez tego konkretnego klienta.

[1] Wzorzec ten został szczegółowo opisany w [20, 101-109].

[2] Szczegółowe omówienie wzorca singleton można znaleźć w [20, 130-138].

sekcja definicji ścieżek danych — zawiera definicję ścieżek danych w programie. Pozwala na definiowanie, do którego odbiorcy danych mają trafić dane odebrane od wskazanego klienta.

Podczas uruchamiania programu plik konfiguracyjny zostaje przeczytany oraz sprawdzony pod kątem poprawności składniowej i semantycznej. Obiekty dostawców, odbiorców oraz algorytmów uwierzytelnienia dostarczane są przez zewnętrznych programistów, zatem prawidłowość ich ustawień nie może być sprawdzana na tym samym etapie. Jest to wykonywane dopiero w trakcie inicjalizacji danego obiektu. Należy zatem zawsze po pomyślnej analizie pliku konfiguracyjnego sprawdzić zawartość pliku dziennika wykonania programu.

Definicje klientów znajdujące się w pliku konfiguracyjnym są niezbędne dla zapewnienia bezpieczeństwa całego systemu. Dodatek NSCAv2 pozwoli na komunikację z klientem, tylko jeśli został on zdefiniowany w konfiguracji. Według wymagań konieczne jest zapewnienie niezależności algorytmów uwierzytelnienia klienta od pozostałych elementów. Definicja każdego klienta zawiera zatem listę modułów uwierzytelnienia, których dany klient może używać. Program zapewnia również kontrolę przesyłanych danych, zatem konieczne jest określenie listy dozwolonych urządzeń i usług, o których dany klient może przesyłać informacje.

Jednym z wielu wymagań stawianych na etapie projektowania systemu było zapewnienie możliwości przekazywania danych pochodzących od klienta mobilnego do wielu miejsc docelowych, takich jak systemy monitorujące czy bazy danych. Polityka dostarczania danych jest zdecydowanie elementem konfiguracyjnym i nie powinna być zaszyta w implementacji programu. W wykonanym programie logika przekazywania danych definiowana jest w programie poprzez sekcję definicji ścieżek danych. Sekcja ta pozwala na definiowanie, w jakie miejsca powinny zostać przekazane dane na podstawie klienta, który te dane przysłał oraz dostawcy, który te dane odebrał.

Przeniesienie znacznej części logiki programu do pliku konfiguracyjnego miało znaczący wpływ na architekturę rozwiązania. Program stanowi jedynie zbiór elementów, z których przy pomocy pliku konfiguracyjnego składa się w pełni funkcjonalny dodatek.

6.2. Szkielet programu

Moduł ten zawiera podstawowe komponenty programu. Zawarto tutaj wszystkie czynności przygotowawcze związane z wczytaniem konfiguracji oraz utworzeniem obiektów z niej wynikających. Ponadto w module tym znajdują się definicje podstawowych bytów logicznych programu. W zależności od pełnionej funkcji można wyróżnić następujące grupy obiektów:

— Grupa obiektów konfiguracyjnych zawiera wszystkie obiekty używane do wczytania parametrów uruchomienia programu z linii poleceń, jak również obiekty odpowiedzialne za dostarczenie do programu konfiguracji zawartej w pliku.
— Grupa obiektów producentów danych zawiera generyczny interfejs producenta danych oraz fabrykę, umożliwiającą pozyskiwanie obiektów z tej grupy, a także definicję dostępnych producentów danych.

— Grupa obiektów konsumentów danych zawiera generyczny interfejs konsumenta danych oraz fabrykę, umożliwiającą pozyskiwanie obiektów z tej grupy, a także definicję dostępnych konsumentów danych.
— Grupa obiektów filtrujących zawiera obiekty pozwalające na kontrolę danych otrzymywanych od klienta.
— Grupa obiektów kanału komunikacyjnego zawiera obiekty powiązane z kanałem komunikacyjnym pomiędzy producentami danych, a ich konsumentami. Zawiera również mechanizmy formatowania danych oraz bufory przeznaczone na dane oczekujące na przekazanie.
— Grupa obiektów zarządzających zawiera zarządcę programu oraz obiekty pomocnicze. Wykonywane są tutaj wszelkie czynności, które muszą wystąpić w trakcie uruchamiania programu. Zachodzi tu także tworzenie oraz niszczenie obiektów implementujących elementy logiczne programu.

Grupa obiektów konfiguracyjnych

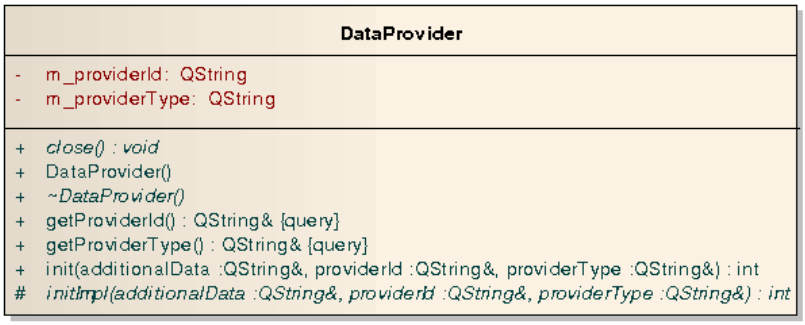
Głównym członkiem grupy obiektów konfiguracyjnych jest parser pliku konfiguracyjnego. Ponieważ plik konfiguracyjny posiada strukturę pliku XML, możliwe było wykorzystanie czytelnika strumienia XML z biblioteki Qt. Klasa ta zapewnia generację znaczników oraz sprawdzanie poprawności składniowej czytanego dokumentu. Umożliwiło to implementację prostego parsera rekursywnie zstępującego, który zajmuje się jedynie sprawdzaniem poprawności logicznej znaczników. Ze względu na strukturę plików XML nie jest możliwa pełna bieżąca kontrola danych w nim zawartych. Konieczne jest zatem wczytanie pliku konfiguracyjnego i odwzorowanie go w strukturach danych, a następnie wykonanie sprawdzenia spójności oraz poprawności tych danych. Obiekt parsera konfiguracji jest również globalnym obiektem udostępniającym parametry konfiguracji. W obiekcie tym znajdują się wszystkie ustawienia oraz definicje wszystkich obiektów logicznych programu.

Grupa obiektów producentów danych

Grupa obiektów producentów danych składa się z dwóch głównych elementów. Pierwszym z nich jest klasa implementująca wzorzec fabryki. Pozwala ona pozostałym obiektom na uzyskiwanie instancji obiektu dostawcy danych, bez konieczności znania jego typu faktycznego w trakcie pisania kodu czy też kompilacji. Drugim z elementów jest generyczny interfejs dostawcy danych, który musi być implementowany przez każdego dostawcę. Interfejs ten jest bardzo prosty jednak pozwala na wykonywanie wszystkich niezbędnych operacji.

Każdy dostawca danych powinien dostarczyć implementacji dwóch funkcji. Funkcja *initImpl()* jest wywoływana z funkcji *init()* w chwili inicjalizacji konkretnego obiektu klasy parametrami pochodzącymi z pliku konfiguracyjnego. Pozwala to między innymi na przekazanie dodatkowych danych inicjujących, takich jak adres sieciowy czy numer portu. Dostawca danych powinien w funkcji *initImpl()* pobrać wszystkie niezbędne mu dane z dostarczonych parametrów, a następnie wykonać wszystkie czynności przygotowawcze takie jak uruchomienie dodatkowego wątku. Po wyjściu z tej funkcji z sukcesem przyjmuje się, ze dostawca został zainicjalizowany poprawnymi danymi oraz jest on uruchomiony i oczekuje na dane od klientów. Istotne jest również dostarczenie implementacji funkcji *close()*. Jest ona odpowiedzialna za zakończenie pracy tego dostawcy danych i zwolnienie wszystkich używanych przez niego zasobów.

Rysunek 6.1. Interfejs dostawcy danych.



Należy również nadmienić, że konieczne było również dostarczenie odpowiedniego mechanizmu rejestracji definiowanych obiektów w fabryce. Istotne jest, aby umożliwić programiście rejestrację nowego typu faktycznego obiektu bez konieczności ingerencji w inne pliki źródłowe. W celu ułatwienia tego procesu zostało opracowane makro, które dzięki wykorzystaniu szablonów dokonuje automatycznej rejestracji nowego typu faktycznego obiektu w fabryce. Dzięki jego wykorzystaniu programista podczas dodawania nowego dostawcy danych, musi zapewnić jedynie deklarację oraz definicję nowego typu. Warto zauważyć, że sposób implementacji tego makra pozwala na umieszczenie definicji nowego typu w pliku nagłówkowym, który jest włączany do wielu jednostek translacji i nie powoduje to błędów kompilacji ani błędów funkcjonowania procesu rejestracji.

Wydruk 6.1. Definicja dostawcy danych

```
DATA_PROVIDER(NazwaTypu, NazwaRejestrowana)
{
  //deklaracja metod
}
```

W ramach tej grupy obiektów dostarczono również referencyjną implementację dostawcy o nazwie typu *DefaultTcpProvider*. Stanowi on implementację omówionego wcześniej protokółu komunikacyjnego. W celu zapewnienia lepszej wydajności logika funkcjonowania tego dostawcy została przeniesiona do osobnego wątku programu. Aby zapewnić elastyczności wykorzystania tego obiektu możliwe jest zarówno definiowanie z poziomu pliku konfiguracyjnego adresu IP i portu wykorzystywanego przez ten obiekt, jak i wskazanie pliku zawierającego klucz prywatny.

Grupa obiektów konsumentów danych

Grupa obiektów konsumentów danych posiada analogiczną budowę jak grupa producentów danych. Zapewnia ona fabrykę konsumentów danych i generyczny interfejs konsumenta. Interfejs konsumenta danych jest analogiczny jak dla dostawcy. Rejestracja obiektów w fabryce odbywa się również przy użyciu analogicznego makra. W ramach tej grupy obiektów dostarczono implementację dwóch konsumentów danych. Pierwszy z nich, o nazwie typu *ToScreenPrinter*, spełnia jedynie funkcję kontrolną. Wszystkie dane, które do niego trafią, są natychmiast

wypisywane w dzienniku wykonania programu. Dostawca typu *ToIcingaWriter* odpowiedzialny jest natomiast za przekazywanie wszystkich danych do pliku komend zewnętrznych systemu *Icinga*. Możliwa jest zmiana ścieżki pliku komend zewnętrznych systemu *Icinga* poprzez plik konfiguracyjny.

Grupa obiektów filtracji

Grupa obiektów filtracji pozwala na kontrolę danych otrzymywanych od klienta, a także na pobieranie informacji o danym kliencie. Podstawowym elementem tej tej grupy jest fizyczne odwzorowanie bytu klienta w odpowiednią klasę. Pozwala to na uzyskiwanie przez inne moduły w prosty sposób wszystkich niezbędnych informacji na temat obsługiwanego klienta. Każdy klient posiada zdefiniowany w pliku konfiguracyjnym zbiór urządzeń i usług, o których informacje może przesyłać. Obiekty z tej grupy, które są odpowiedzialne za kontrolę odbieranych danych, pobierają z analizatora składniowego wspomniane zbiory i dokonują kontroli każdego wpisu dziennika otrzymanego od klienta. Jeśli klient nadesłał dane dotyczące urządzenia lub usługi, do których nie ma on uprawnień, nie będzie możliwe przekazanie ich do konsumentów.

Grupa obiektów kanału komunikacyjnego

Grupa obiektów kanału komunikacyjnego odpowiedzialna jest za niezawodne przekazanie danych od dostawcy danych do konsumentów według reguł zdefiniowanych w pliku konfiguracyjnym. Zapewnienie niezawodności zostało osiągnięte poprzez implementację bufora kołowego wewnątrz pliku. Każdy dostawca danych posiada swój plik bufora. Na początku tego pliku zapisane są położenia miejsca przeznaczonego do czytania oraz miejsca przeznaczonego do pisania. Dane, które logicznie znajdują się pomiędzy miejscem do czytania, a miejscem do pisania, są to dane, które zostały odebrane od klienta, lecz nie zostały jeszcze dostarczone do konsumentów. Pomyślnie zweryfikowana przez obiekty filtrujące porcja danych przekazywana jest z użyciem kanału komunikacyjnego do konsumentów danych. Operacja ta odbywa się w dwóch etapach. Pierwszy etap dokonuje zapisu danych do pliku bufora. Jeśli aktualnie nie ma żadnych danych oczekujących na przetworzenie przez konsumentów, nowa porcja danych jest niezwłocznie dostarczana do odpowiednich obiektów. Jeśli istnieją porcje danych, które oczekują na zapisanie, dane zostaną zapisane do pliku i dostarczone po danych, które nadeszły przed nimi. Należy zwrócić uwagę, że dane zapisywane są na dysku tylko raz, niezależnie od liczby konsumentów, do których powinny one zostać dostarczone. Ponadto dostawca danych uzyskuje potwierdzenie zapisania danych po zakończeniu pierwszego etapu ich obsługi, czyli już po zapisaniu do pliku bufora. Oznacza to, że dostawcy danych są w znacznym stopniu niezależni od konsumentów danych. Pozwala to skrócić do minimum czas oczekiwania klienta mobilnego pomiędzy wysłaniem danych, a uzyskaniem potwierdzenia o ich przetworzeniu.

Grupa obiektów zarządzających

Głównym przedstawicielem grupy obiektów zarządzających jest klasa główna programu. Program został napisany zgodnie z metodyką obiektową, zatem cała logika wykonania programu została również zamknięta w klasie, co uprościło funkcję główną programu do minimum. Klasa ta odpowiedzialna jest za przebieg całości programu. Pierwszą operacją wykonywaną przez tę klasę jest odnalezienie pliku konfiguracyjnego i zlecenie jego wczytania przez parser konfiguracji. Na podstawie

informacji uzyskanych w wyniku analizy pliku konfiguracyjnego klasa główna programu pobiera z odpowiednich fabryk wszystkie obiekty producentów oraz konsumentów zdefiniowanych w pliku konfiguracyjnym. Po uzyskaniu obiektów następuje ich inicjalizacja na podstawie danych wczytanych z pliku konfiguracyjnego. Klasa ta jest również odpowiedzialna za prawidłową deinicjalizację oraz destrukcję wszystkich obiektów. Ponadto należy zauważyć, że omawiany program funkcjonuje jako serwis systemowy, zatem konieczne jest również wykonanie w tej klasie wszystkich czynności zalecanych przy uruchamianiu takich serwisów. Całość programu została napisane zgodnie z paradygmatem zdarzeniowym. Oznacza to, że po wykonaniu inicjalizacji, program zawiesza swoje wykonanie do czasu otrzymania jakiegoś zdarzenia, na które powinien on zareagować. Użycie biblioteki Qt w znaczny sposób uprościło oczekiwanie na zdarzenia dzięki możliwości użycia pętli zdarzeń z tej biblioteki.

6.3. Moduł kryptograficzny

Moduł ten dostarcza pozostałym elementom programu implementacji algorytmów kryptograficznych. Dostępne są generyczne interfejsy do następujących schematów algorytmów:

— szyfrowanie symetryczne,
— szyfrowanie asymetryczne,
— funkcja skrótu,
— podpis cyfrowy.

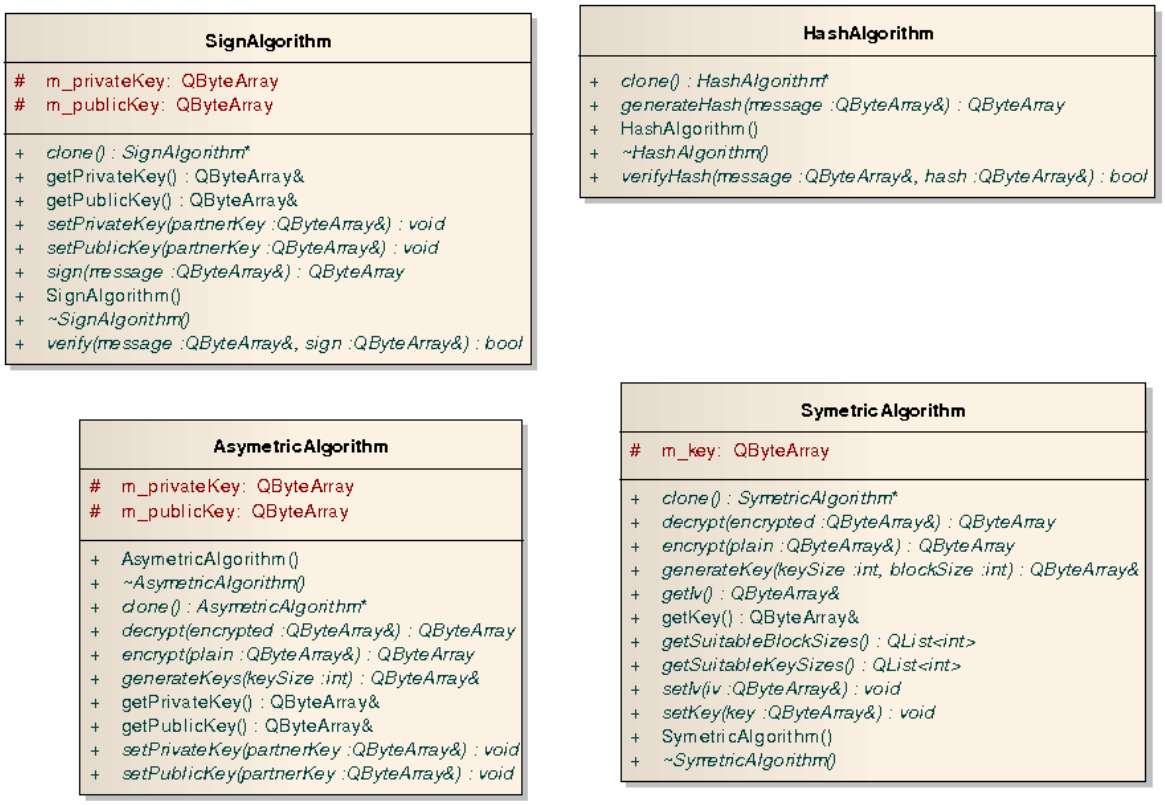
Definicje wszystkich interfejsów zostały przedstawione na rys. 6.2. Dostarczanie implementacji danego schematu kryptograficznego odbywa się w sposób analogiczny do dostarczania implementacji dostawców danych. Wszystkie implementacje algorytmów zostają zarejestrowane w fabryce kryptograficznej, która umożliwia uzyskiwanie obiektu o typie określanym na podstawie danych, na przykład odebranych od klienta.

W ramach tej pracy dostarczono kilku algorytmów, które były konieczne do zaimplementowania protokołu komunikacyjnego. Jako algorytm symetryczny dostarczona została implementacja algorytmu AES pracującego w trybie wiązania bloków zaszyfrowanych. Omawiany tryb pracy algorytmu powoduje powstanie zależności pomiędzy kolejnymi blokami. Oznacza to, że manipulacja jednym blokiem powoduje zmiany wartości odszyfrowanych w tym bloku i każdym następnym. Zastosowanie tego trybu w implementowanym protokole komunikacyjnym pozwala na zagwarantowanie, że sekwencja wiadomości otrzymywanych od klienta nie została zmieniona[3]. Protokół komunikacyjny wymagał również dostarczenia asymetrycznego algorytmu RSA. W module zdefiniowano ponadto klasę implementującą generyczny interfejs funkcji skrótu, która dostarcza funkcjonalności algorytmu SHA-2 o długości skrótu 256 bitów. Jeden z etapów protokołu komunikacyjnego wymagał również dostarczenia algorytmu podpisu cyfrowego opartego na algorytmie RSA.

Do implementacji wszystkich algorytmów została wykorzystana biblioteka Crypto++. Jest to popularna biblioteka o otwartych źródłach, napisana w języku C++, która w obiektowy sposób udostępnia algorytmy kryptograficzne.

[3] Samo wykorzystanie trybu CBC nie daje gwarancji ale w protokole komunikacyjnym użyto również funkcji skrótu, przez co możliwe jest jej udzielenie.

Rysunek 6.2. Interfejsy algorytmów kryptograficznych.



6.4. Moduł uwierzytelnienia klienta

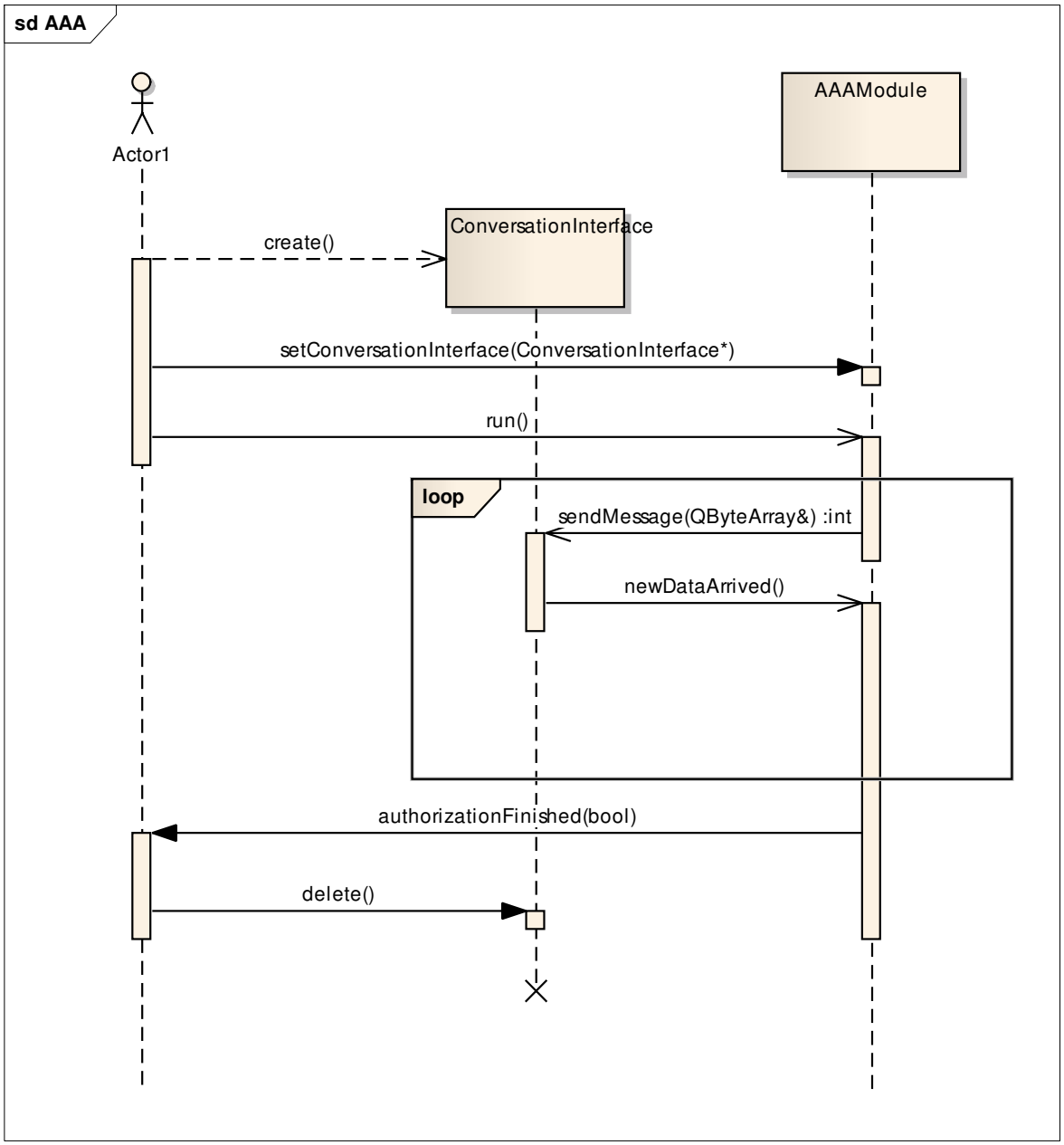
Moduł posiada architekturę typową dla modułów usługowych. Głównym elementem jest klasa implementująca wzorzec fabryki oraz generyczny interfejs pozwalający na wykorzystywanie obiektów uzyskanych z fabryki.

Interfejs zdefiniowany dla algorytmów uwierzytelnienia został zaprojektowany z wykorzystaniem mechanizmu sygnałów i slotów z biblioteki Qt. Użycie tego mechanizmu pozwala na znacznie wydajniejsze wykorzystanie zasobów. Przykładem takiej optymalizacji może być czas, gdy klient przetwarza żądanie związanie z uwierzytelnieniem. Wątek serwera nie musi być wtedy bezczynny, lecz może przetwarzać żądania pochodzące od innych klientów. Definicja interfejsu pozwala również na przekazanie do niego dodatkowych ustawień pochodzących z pliku konfiguracyjnego. Każdy algorytm uwierzytelnienia powinien po zakończeniu sukcesem lub porażką procesu uwierzytelnienia klienta wykonać emisję sygnału z interfejsu algorytmu wraz z rezultatem procesu autoryzacji. Ponieważ z punktu widzenia protokołu istotne jest przekazanie danych autoryzacyjnych jako informacji o akceptacji algorytmu uwierzytelnienia, konieczne jest, aby każdy algorytm przesłał co najmniej jedną wiadomość do klienta.

Zapewnienie niezależności implementacji algorytmów uwierzytelnienia od wykorzystywanej aktualnie metody komunikacji wymagało zdefiniowania generycznego interfejsu komunikacyjnego, który może być wykorzystywany przez implementacje

poszczególnych algorytmów. Implementacja tego interfejsu powinna być dostarczona przez moduł, który jest aktualnie wykorzystywany w programie do komunikacji z klientem.

Rysunek 6.3. Przykładowy przebieg procesu uwierzytelnienia.



Przykładowy przebieg procesu uwierzytelnienia został przedstawiony na rys. 6.3. Proces ten rozpoczyna się od ustawienia aktualnie wykorzystywanego interfejsu komunikacyjnego. Następnie wykonywane jest rozpoczęcie procesu uwierzytelnienia. W ramach tego procesu moduł uwierzytelnienia może komunikować się z klientem poprzez otrzymany wcześniej obiekt. Po zakończeniu procesu autoryzacji emitowany jest odpowiedni sygnał informujący pozostałe komponenty o rezultacie uwierzytelnienia.

W ramach pracy został zaimplementowany również przykładowy moduł uwierzytelnienia klienta. Nosi on nazwie: *LoginPass*. Jest to prosta metoda uwierzytelnienia oparta na pobraniu od klienta loginu oraz hasła i porównanie go z danymi dostarczonymi w pliku konfiguracyjnym. Każdy klient posiada w pliku konfiguracyjnym listę dostępnych dla niego algorytmów uwierzytelnienia wraz z danymi, jakie powinny być przekazane, aby zapewnić pozytywne wykonanie procesu. Należy zwrócić uwagę, że zaimplementowany algorytmy uwierzytelnienia stanowi jedynie przykład, w jaki sposób powinno się definiować algorytmy w opracowanym dodatku. Ich wykorzystanie w systemie produkcyjnym zależy od bezpieczeństwa serwera, na którym znajduje się program NSCAv2. Należy być świadomym, że wszystkie hasła oraz nazwy użytkowników przechowywane są jawnym tekstem w pliku konfiguracyjnym[4]. Nie stanowi to oczywiście zagrożenia jeśli do serwera monitorującego dostęp mają jedynie uprzywilejowane osoby. Należy przypomnieć, że na serwerze znajdują się również klucze prywatne RSA, więc zawsze należy zwrócić szczególną uwagę, na bezpieczeństwo serwera.

6.5. Moduł komunikacji z wykorzystaniem TCP

Moduł ten zawiera implementację protokołu komunikacyjnego opisanego w rozdziale 5.2. Inicjacja każdej z warstw protokołu została zaimplementowana w dedykowanej klasie lub jeśli proces inicjacji złożony był z kilku rozdzielnych logicznie elementów, każdy element został zaimplementowany w osobnej klasie. W celu umożliwienia każdej z warstw protokołu korzystanie z usług warstw niższych, w sposób generyczny wykorzystany został wzorzec dekoratora.

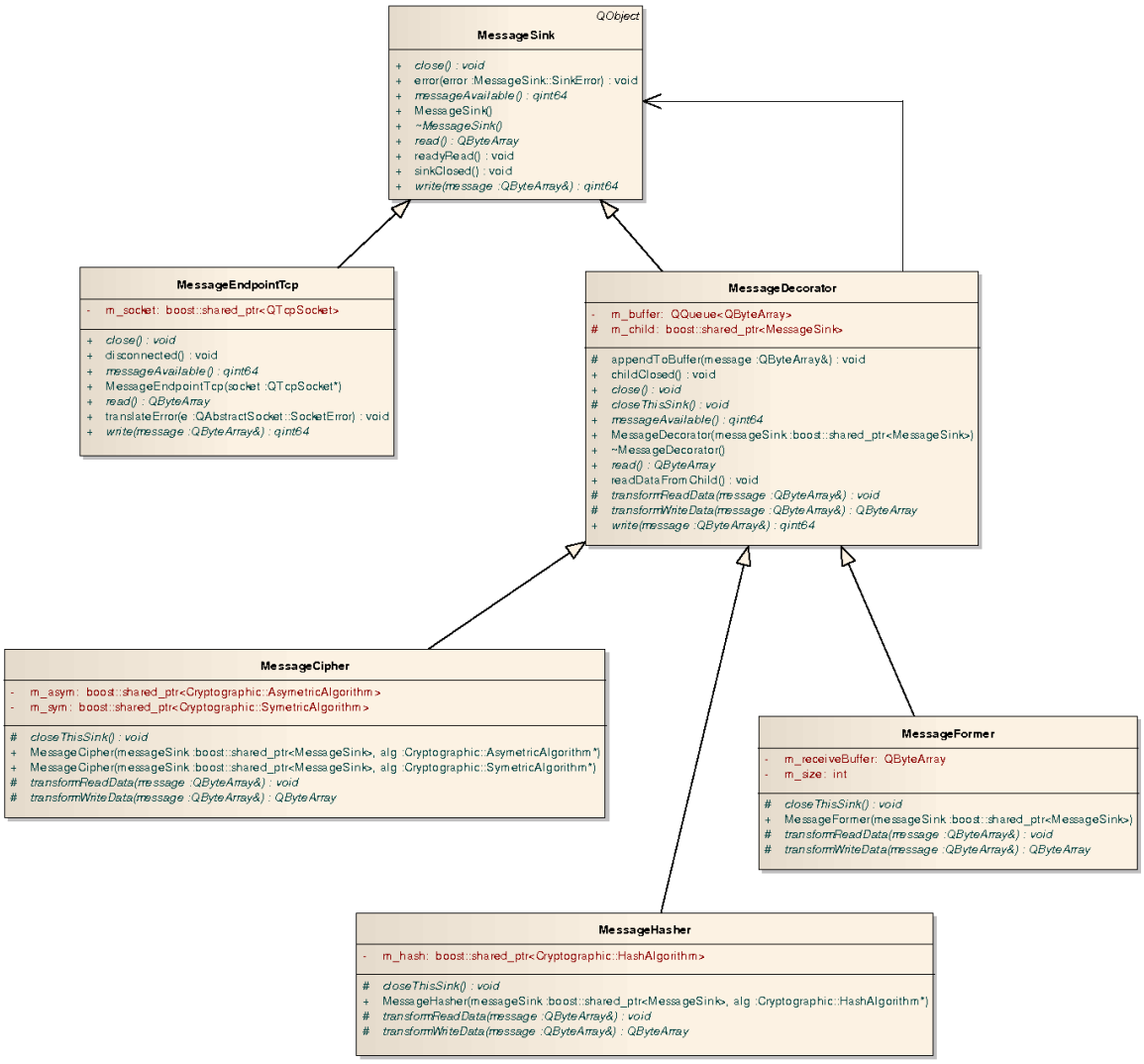
Aby wykorzystać wzorzec dekoratora (patrz rys. 6.4), zdefiniowano generyczny interfejs pozwalający na odczytanie oraz zapisanie komunikatu niezależnie od liczby warstw znajdujących się poniżej. Klasą prostą w tym przypadku jest prosta klasa opakowująca gniazdo TCP z biblioteki Qt. Klasami dekorującymi są klasy odpowiadające za kolejne czynności w wyższych warstwach protokołu. Klasy te są odpowiedzialne za formowanie wiadomości, szyfrowanie oraz obliczanie i kontrolę jej skrótu. Ponieważ kolejność czynności wykonywanych w trakcie budowania wiadomości jest istotna, zastosowano enkapsuowane budowanie wiadomości. Każda klasa dekoratora posiada tylko jedno wskazanie do obiektu, który znajduje się o poziom niżej w hierarchii. Użytkownik zapisuje wiadomość, używając klasy z najwyższej warstwy. Obiekt ten dokonuje przekształcenia wiadomości zgodnie ze swoim algorytmem, a następnie wywołuje tę samą metodę na rzecz obiektu znajdującego się o jeden niżej w hierarchii niż ona sama, przekazując przekształconą wiadomość jako parametr wywołania.

Wykorzystanie wzorca dekoratora pozwoli w przyszłości na łatwą modyfikację protokołu komunikacyjnego, np. poprzez dodatkową warstwę. Ponadto wprowadzenie jednolitego interfejsu pozwoliło na zachowanie prostoty i jednolitości implementacji poszczególnych warstw protokołu komunikacyjnego.

Moduł ten został zaimplementowaniu przy użyciu licznych mechanizmów z biblioteki Qt. Przede wszystkim wykorzystany został moduł sieciowy wspomnianej biblioteki. Dzięki jego użyciu uzyskano dostęp do generycznej implementacji serwera TCP, a także gniazd. Szkielet aplikacji Qt pozwolił na wygodną implementację

[4] Sposób przechowywania loginu oraz hasła na urządzeniu mobilnym jest uzależnione od konkretnej implementacji aplikacji IcingaMini.

Rysunek 6.4. Diagram klas wykorzystujących wzorzec dekoratora.



asynchronicznej komunikacji z użyciem gniazd TCP przy pomocy standardowego dla tego szkieletu mechanizmu sygnałów i slotów. Dzięki temu uzyskano przejrzysty i wydajny kod, który pozwala na obsługę wielu klientów w jednym wątku.

6.6. Moduł logowania

Omawiany dodatek wykonywany jest bez interakcji z użytkownikiem. Funkcjonuje on jako serwis systemowy. Docelowo będzie wykonywany na serwerze, poza sesją jakiegokolwiek użytkownika. W trakcie wykonania programu mogą się zdarzyć sytuacje wymagające poinformowania użytkownika o ich wystąpieniu. Znaczna część z tych informacji stanowi jedynie zapis wykonania programu, jednak mogą występować również informacje o sytuacjach krytycznych, o których użytkownik musi zostać powiadomiony. Konieczne było zatem dostarczenie możliwości przekazywania takich informacji z wielu modułów do jednego, wspólnego miejsca, które stanowi dziennik wykonania serwisu.

Każdy moduł posiada możliwość przekazywania użytkownikowi wiadomości o różnym priorytecie. Dozwolone są następujące priorytety:

FATAL — najwyższy priorytet, wiadomość zawiera komunikat o błędzie, który uniemożliwia dalsze wykonanie programu.
ERROR — komunikat zawiera informacje o błędzie, który uniemożliwia wykonanie pewnej ścieżki programu.
WARNING — komunikat zawiera ostrzeżenie o nietypowej sytuacji.
DEBUG — komunikat zawiera treść pomocną podczas wyszukiwania błędów.
INFO — komunikat zawiera jedynie treści informacyjne.

Podczas kompilacji ustalany jest minimalny priorytet wiadomości, które mają być przekazywane użytkownikowi. Wszystkie wiadomości o priorytecie niższym niż ustalony nie zostaną zapisane. Ponadto dzięki użyciu mechanizmów opartych o szablony wszystkie komunikaty o priorytecie niższym zostaną rozwinięte do wywołania funkcji pustej. Wywołanie takie zostanie z bardzo dużym prawdopodobieństwem zoptymalizowane przez kompilator.

Przekazanie użytkownikowi treści komunikatu w wielu sytuacjach może nieść zbyt mało informacji. Aby umożliwić przekazanie dodatkowych informacji bez konieczności pisania nadmiernej liczby komend przy każdym komunikacie, opracowana została makrodefinicja, która do każdego komunikatu dołączy aktualny stempel czasu, nazwę pliku, w którym znajduje się komunikat, a także nazwę funkcji oraz numer linii. Ponadto komunikat nie musi się składać jedynie z tekstu, lecz można go formować w taki sam sposób jak pisać do strumienia.

Wydruk 6.2. Przykładowe wypisanie komunikatu.

```
LOG_ENTRY(MyLogger::DEBUG, "komunikat"<<123);
```

Wydruk 6.3. Format komunikatu przekazywanego użytkownikowi.

```
[stempel czasu][poziom][plik][funkcja][linia]:komunikat123
```

Ponieważ demon nie jest przypisany do żadnego z terminali[5], nie ma możliwości przekazywania wiadomości na standardowe wyjście lub wyjście błędów. Konieczne jest zatem utworzenie pliku, do którego zapisywane będą komunikaty. Należy zwrócić uwagę, że program jako serwis systemowy uruchomiony będzie ze znacznie ograniczonymi prawami, aby podnieść poziom bezpieczeństwa serwera. W związku z powyższym jedynym miejscem, co do którego można założyć, że program będzie miał dostęp, jest katalog plików tymczasowych. Każde uruchomienie programu powoduje zatem utworzenie w tym katalogu pliku składającego się z nazwy programu oraz stempla czasu zawierającego czas jego uruchomienia.

[5] Proces przekształcenia w serwis systemowy zakłada zamknięcie deskryptorów standardowego wejścia, wyjścia oraz wyjścia błędów.

7. Doświadczenia praktyczne z pracy z systemem

W ramach tej pracy wykonano prototypową instalację i konfigurację rozbudowanego systemu *Icinga*, która została uruchomiona w niewielkiej sieci domowej. W ramach instalacji uruchomiono również opracowany dodatek NSCAv2, do którego z powodzeniem podłaczono dwa urządzenia mobilne pod kontrolą aplikacji monitorującej IcingaMini [22]. Pozwoliło to na uruchomienie kompletnego systemu oraz weryfikację poprawności jego implementacji i zdefiniowanych wymagań.

7.1. Opis infrastruktury

Konfiguracja systemu Icinga (wraz z szeregiem dodatków) została wykonana w mieszkaniu autora pracy w oparciu o istniejącą infrastrukturę sieci lokalnej. Sercem konfiguracji jest laptop HP Compaq 6710b. Został na nim zainstalowany system operacyjny Linux. Wybrano najnowszą dostępną w czasie wykonywania konfiguracji, wersję dystrybucji Ubuntu — 13.10. System został zainstalowany w wersji 64-bitowej. W celu zapewnienia możliwości łatwego przeniesienia wykonanej konfiguracji systemu monitorującego, utworzono na tym serwerze maszynę wirtualną. Do jej stworzenia użyto narzędzi opartych o bibliotekę *libvirt*. Maszyna wirtualna uruchamiana jest jako KVM[1], co zapewnia jej wysoką wydajność. Na utworzonej maszynie wirtualnej zainstalowany został ten sam system operacyjny, co na urządzeniu będącym jej gospodarzem. Interfejs sieciowy urządzenia został skonfigurowany w taki sposób, aby urządzenie fizyczne i maszyna wirtualna widziane były z zewnątrz jako dwa interfejsy o różnych adresach MAC.

Poza laptopem oraz maszyną wirtualną w sieci dostępne są również inne urządzenia. Pierwszym z tych urządzeń jest drukarka sieciowa Samsung CLP-610ND. Posiada ona interfejs sieciowy, na którym udostępniana jest usługa bezpośredniego drukowania i administracji poprzez stronę internetową z użyciem protokołu HTTP. Drugie urządzenie to router ASUS WL-500GPv2. Jest to element odpowiedzialny za przydział adresów w całej sieci z użyciem protokołu DHCP. Ponadto jest on bramą domyślną dla wszystkich urządzeń znajdujących się w sieci lokalnej. Wyposażony jest w interfejs zarówno przewodowy, jak i bezprzewodowy. Urządzenie to udostępnia panel administracyjny w formie strony internetowej poprzez protokół HTTP. Ponadto w sieci znajduje się przełącznik ASUS GigaX 1005/G, dzięki któremu urządzenia połączone są w sieć lokalną bez konieczności korzystania z routera. W dalszych rozważaniach urządzenie to zostało pominięte, ponieważ jest to prosty przełącznik warstwy drugiej i nie jest możliwe jego monitorowanie.

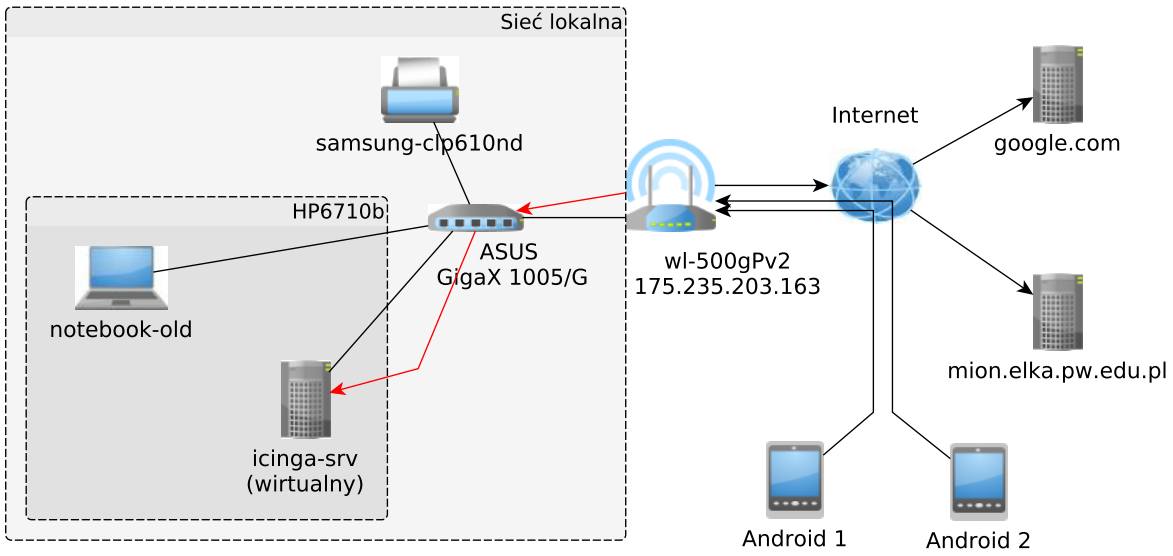
[1] ang. *Kernel Virtual Machine* – sposób wirtualizacji w systemach Linuksa, gdzie proces ten jest wspierany przez jądro systemu gospodarza

Testowa sieć lokalna zawiera zdecydowanie zbyt mało elementów, aby móc w pełni przetestować działanie systemu. Ponadto jest ona w znacznym stopniu odizolowana od czynników zewnętrznych mogących wpływać na pracę systemu monitorującego. W celu wykonania bardziej realistycznych testów postanowiono monitorować dwa dostępne publicznie serwery. Pierwszym z nich jest serwer *google.com*. Drugi natomiast to serwer *mion.elka.pw.edu.pl* przeznaczony dla studentów Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej.

W ramach niniejszej pracy wykonano rozbudowę systemu monitorującego *Icinga* o funkcjonalność monitorowania klienta mobilnego. W związku z powyższym konieczne było uwzględnienie również klientów mobilnych w konfiguracji testowej. Dzięki uprzejmości Pana Marcina Kubika możliwe było wykorzystanie opracowanego przez niego modułu mobilnego dla platformy Android (smartfon Samsung Galaxy Note 3 oraz Samsung Galaxy S2 Plus). Pozwoliło to na dodanie do monitorowanej infrastruktury dwóch urządzeń mobilnych pracujących pod kontrolą tego systemu. Oba urządzenia były używane do codziennych czynności, co pozwoliło na symulacje normalnych warunków pracy systemu.

Aby umożliwić komunikację aplikacji mobilnej z dodatkiem NSCAv2 spoza sieci domowej, konieczne było użycie mechanizmu przekazywania portów na wspomnianym wcześniej routerze. Schemat wykonanej testowej infrastruktury przedstawiono na rysunku 7.1. Na schemacie zachowano porządek oznaczeń urządzeń przedstawiony w tabeli 7.1.

Rysunek 7.1. Schemat infrastruktury testowej. Kolorem czerwonym zostało wyróżnione połączenie wynikające z przekierowania portów.



7.2. Konfiguracja systemu monitorowania

Monitorowana infrastruktura jest dość prosta i mała. Pozwala to na użycie jednego rdzenia monitorującego zarówno do monitorowania infrastruktury statycznej, jak i do przetwarzania danych pochodzących od klientów mobilnych. W celu zapewnienia konfiguracji zbliżonej do warunków użytkowania systemu, wykonano

Nazwa	Opis
notebook-old	System zainstalowany natywnie na laptopie HP 6710b.
icinga-srv	System uruchomiony na maszynie wirtualnej na urządzeniu notebook-old.
samsung-clp610nd	Drukarka Samsung CLP-610ND znajdująca się w sieci lokalnej
wl-500gPv2 175.235.203.163	Router ASUS WL-500GPv2 o zewnętrznym adresie IP 175.235.203.163
google.com	Serwer popularnej wyszukiwarki internetowej Google.
mion.elka.pw.edu.pl	Serwer mion zarządzany przez WEiTI PW.
Android 1	Telefon Samsung Galaxy Note 3.
Android 2	Telefon Samsung Galaxy S2 Plus.

Tablica 7.1: Objaśnienia nazw urządzeń wykorzystywanych w infrastrukturze.

konifguracje wraz z dodatkien inGraph oraz bazą danych dla systemu *Icinga*. Konfiguracja całego systemu składa się z następujących elementów:

— rdzeń monitorujący *Icinga*,
— baza danych MySQL systemu *Icinga*,
— dodatek inGraph,
— baza danych MySQL dodatku inGraph,
— interfejs icinga-web,
— baza danych MySQL icinga-web[2],
— wykonany w tej pracy dodatek NSCAv2,
— aplikacja mobilna dla platformy Android, napisana przez Pana Kubika,
— zestaw wtyczek.

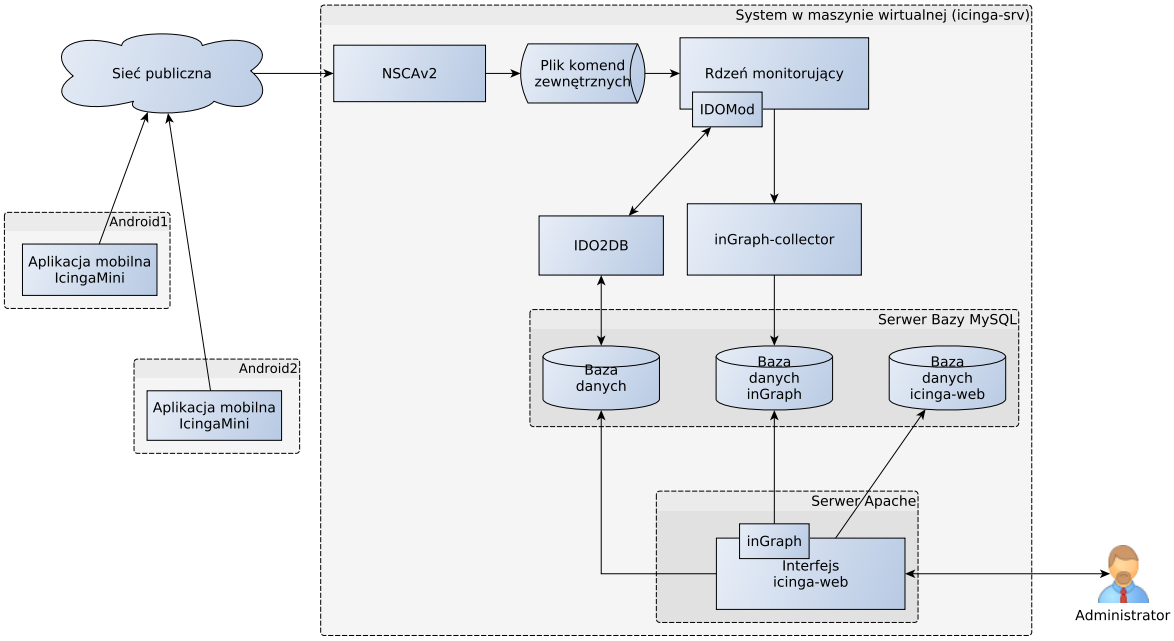
Schemat rozmieszczenia oraz współpracy poszczególnych elementów został przedstawiony na rysunku 7.2. System monitorowania do poprawnego funkcjonowania potrzebuje zarówno serwera bazy danych MySQL, jak i serwera http. W wykorzystywanej wersji dystrybucji Ubuntu wymagane pakiety zostały zainstalowane domyślnie podczas instalacji systemu.

Rdzeń systemu monitorowania został skonfigurowany tak, aby monitorować w sposób aktywny wszystkie usługi klientów statycznych. Do monitorowania użyto zestawu standardowych wtyczek rozwijanych pierwotnie dla systemu Nagios. Lista monitorowanych usług dla każdego urządzenia została przedstawiona w formie tabeli 7.2. Niektóre z monitorowanych wartości wymagają użycia dodatkowego narzędzi - NRPE[3]. Jest to narzędzie, które pozwala na uruchomienie wtyczki na zdalnej maszynie. Jest on niezbędny do pomiaru parametrów wewnętrznych danego urządzenia, takich jak zużycie procesora czy pamięci. Użyto go do minitorowania systemu gospodarza (laptop).

[2] Ze względu na użycie nowego interfejsu konieczna jest dodatkowa baza danych, w której przechowywane są dane warstwy prezentacji. Baza ta nie została wymieniona w opisie, gdyż nie zawiera ona danych o stanie sieci, a jedynie dane interfejsu graficznego.

[3] ang. *Nagios Remote Plugin Executor* – narzędzie do zdalnego uruchamiania wtyczek. Dokładny opis można znaleźć w [13].

Rysunek 7.2. Diagram rozmieszczenia i współpracy elementów systemu monitorowania.



W celu umożliwienia monitorowania klienta mobilnego konieczne było również zdefiniowanie w systemie *Icinga* urządzeń Android1 oraz Android2, a także odpowiednich usług. Udostępniona przez Pana Marcina Kubika wersja aplikacji mobilnej posiada duży zbiór parametrów urządzenia, które można monitorować. Spośród dostępnych parametrów wybrano następujące:

— siła najmocniejszego sygnału Wi-Fi,
— stan baterii,
— liczba uruchomionych aplikacji,
— obciążenie procesora.

Konieczna była również konfiguracja dodatku NSCAv2. Ponieważ telefony posiadały różnych użytkowników, zostały utworzone dwa konta klientów. Jako jedyną dopuszczalną metodę uwierzytelnienia wybrano login oraz hasło. W celu zapewnienia transportu danych konieczne było również nadanie utworzonym użytkownikom uprawnień dotyczących urządzeń i usług. W celu umożliwienia przesłania danych od klienta konieczna była konfiguracja dostawcy danych. Pierwszym jej etapem było wygenerowanie pary kluczy RSA oraz umieszczenie klucza prywatnego i publicznego na serwerze, a publicznego na urządzeniach mobilnych. Konieczne było również podanie w pliku konfiguracyjnym dodatku NSCAv2 ścieżki do klucza prywatnego, a także numeru portu na którym powinien on czekać na przychodzące połączenia. Ponieważ plik komend zewnętrznych w wykonanej instalacji systemu *Icinga* znajduje się w standardowym miejscu, konieczna była jedynie bezparametrowa deklaracja konsumenta danych. Ostatnim etapem konfiguracji było zdefiniowanie ścieżki danych dla klientów, prowadzącej od dostawcy danych do odbiorcy.

Przedstawiona na rys. 7.1 topologia sieci wyraźnie pokazuje, że istnieje w sieci urządzenie, którego awaria może spowodować utratę łączności z wieloma urządzeniami. Elementem tym jest router wl-500gpv2. Awaria tego urządzenia powoduje,

Wartość mierzona	icinga-srv	notebook--old	samsung--clp610nd	wl-500gPv2	google	mion
Liczba procesów	Tak	Tak				
Użycie przestrzeni wymiany	Tak					
SSH	Tak	Tak				Tak
Użycie dysku	Tak					
HTTP	Tak	Tak	Tak	Tak	Tak	Tak
Liczba zalogowanych użytkowników	Tak	Tak				
Bieżące obciążenie	Tak	Tak				
Ping	Tak	Tak	Tak	Tak	Tak	Tak
Liczba procesów	Tak					
IMAP z SSL						Tak
POP3 z SSL						Tak
SMTP						Tak
Liczba procesów zombie		Tak				

Tablica 7.2: Monitorowane usługi i parametry klientów statycznych

że serwer icinga traci możliwość wykonywania sprawdzeń zarówno tego urządzenia, jak i wszystkich urządzeń znajdujących się poza siecią lokalną. W celu ograniczenia liczby komunikatów otrzymywanych w przypadku takiej awarii konieczne było zdefiniowanie odpowiedniej struktury sieci. Rysunek 7.3 przedstawia logiczną strukturę połączeń. Została ona wygenerowane przez program *Icinga* na podstawie plików konfiguracyjnych monitorowanych urządzeń. Łatwo zauważyć, że jedyna ścieżka od systemu monitorującego do wszystkich urządzeń spoza sieci lokalnej prowadzi przez router wl-500gpv2. Odpowiada to oczywiście fizycznym zależnościom w sieci i umożliwi precyzyjną diagnozę ewentualnej awarii.

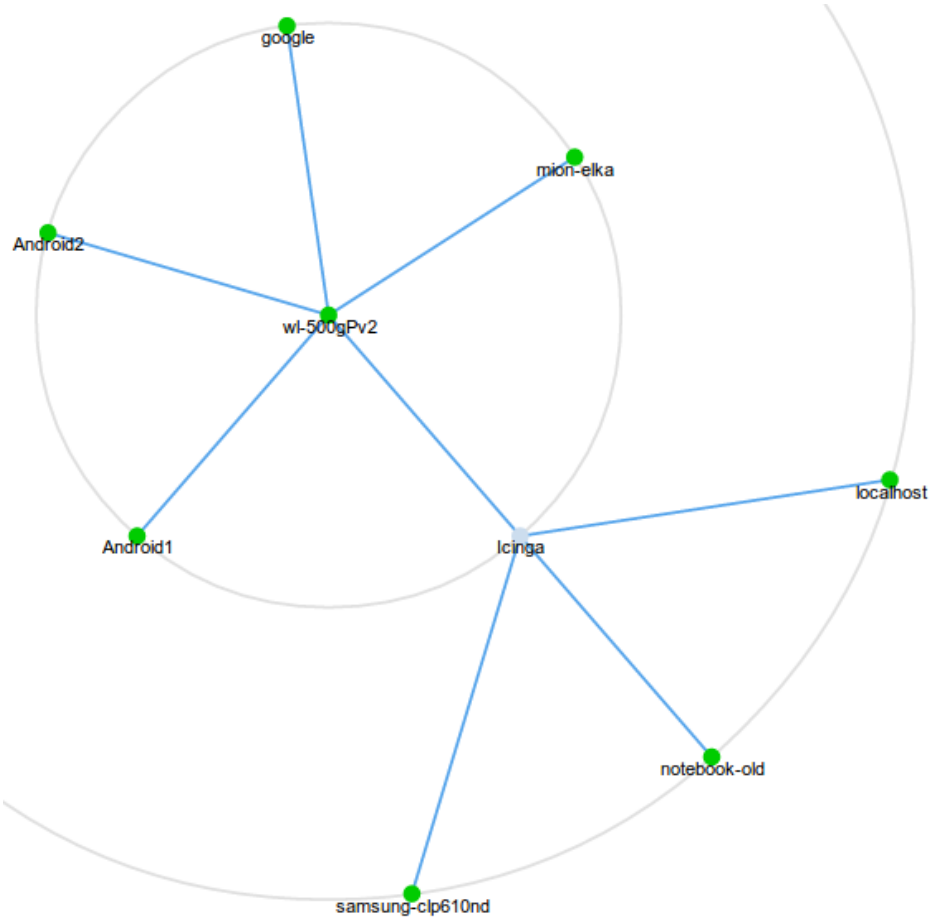
7.3. Rezultaty dla klientów statycznych

Monitorowanie infrastruktury statycznej zostało przeprowadzone w okresie trzech tygodni. Czas ten jest zdecydowanie wystarczający, aby zgromadzić dane, będące wiarygodnym źródłem informacji o sieci. Należy zauważyć, że system monitorowania działał przez cały ten czas bez żadnej awarii.

7.3.1. Sieć lokalna

Testowanie monitorowania sieci lokalnej przebiegło zgodnie z oczekiwaniami. Zgromadzone dane wykazały stałą jakość usług świadczonych w sieci lokalnej. Ze względu na niski poziom skomplikowania infrastruktury testowej, otrzymywane

Rysunek 7.3. Logiczny schemat monitorowanej infrastruktury.

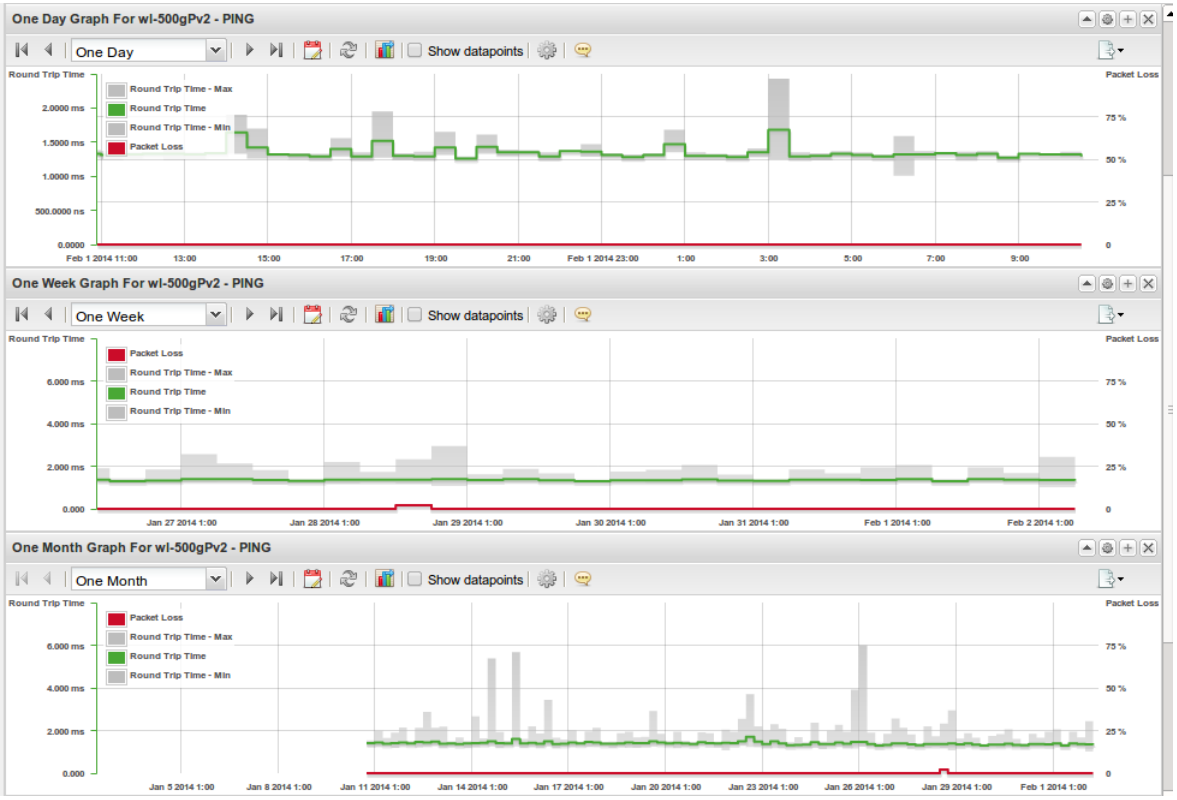


wartości czasu od wysłania pakietu do jego powrotu były rzędu pojedynczych milisekund. Dzięki wykorzystaniu dodatku inGraph możliwe było przedstawienie zgromadzonych danych w formie wykresów.

Przykładowy wykres czasu podróży pakietów oraz poziomu utraconych podczas komunikacji pakietów dla router wl-500gPv2 przedstawiono na rys. 7.4. Na wykresie kolorem zielonym zostały zaznaczone średnie czasy podróży dla zadanych przedziałów. Kolor szary prezentuje natomiast wartości minimalne i maksymalne dla bieżących przedziałów agregacji. Kolor czerwony pokazuje udział utraconych pakietów wobec wszystkich przesłanych.

Przedstawiony wykres potwierdza prawidłowe wyniki przeprowadzonych testów systemu. Średni czas podróży pakietu wynosi poniżej 2 ms. Uzyskane wartości maksymalne są rzędu 6 ms, co również jest zjawiskiem normalnym w sieciach komputerowych, ze względu na zmienne obciążenie routera. Należy zauważyć, że na wykresie wystąpił chwilowy wzrost stopnia utraconych pakietów w okolicach 29 stycznia. Dodatek inGraph pozwala administratorowi, po zauważeniu takiej sytuacji wygenerować wykres dokładniejszych wartości we wskazanym okresie. Funkcja szybkiego widoku, która pozwala na zaznaczenie interesującego obszaru pozwoliła na łatwe przedstawienie dokładnych danych z okresu wystąpienia wzrostu mierzonej wartości. Wykres ten został przedstawiony na rysunku 7.5. Na podstawie tego wykresu można określić dokładny czas wystąpienia interesującego zdarzenia.

Rysunek 7.4. Wykres wartości czasu podróży pakietu oraz stopnia utraconych pakietów dla wl-500gPv2. Wykresy przedstawiają odpowiednio okres jednego dnia, tygodnia oraz miesiąca.



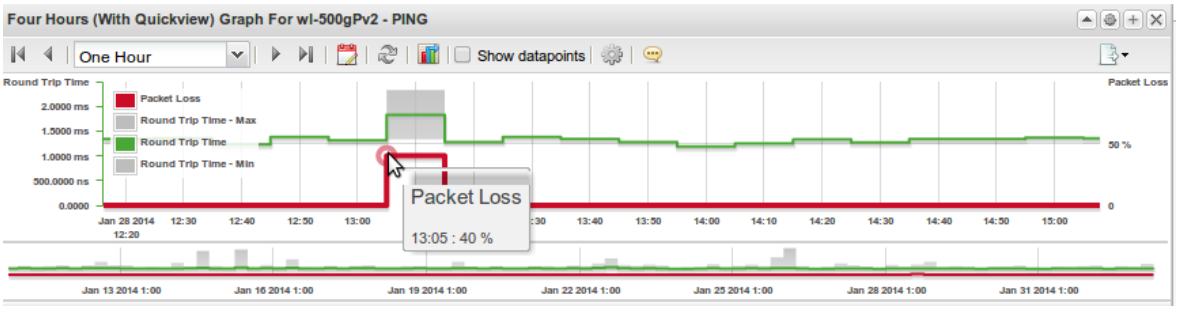
W przypadku wystąpienia usterki może to zostać wykorzystane do diagnozowania jej przyczyny poprzez badanie zdarzeń tuż przed jej wystąpieniem.

7.3.2. Serwery zewnętrzne

Sieć lokalna jest wykorzystywana jedynie przez wąskie grono użytkowników. Powoduje to niewielką zmienność monitorowanych w niej parametrów. Dzięki monitorowaniu serwerów zewnętrznych możliwe było przetestowanie systemu w dużo bardziej realistycznych warunkach.

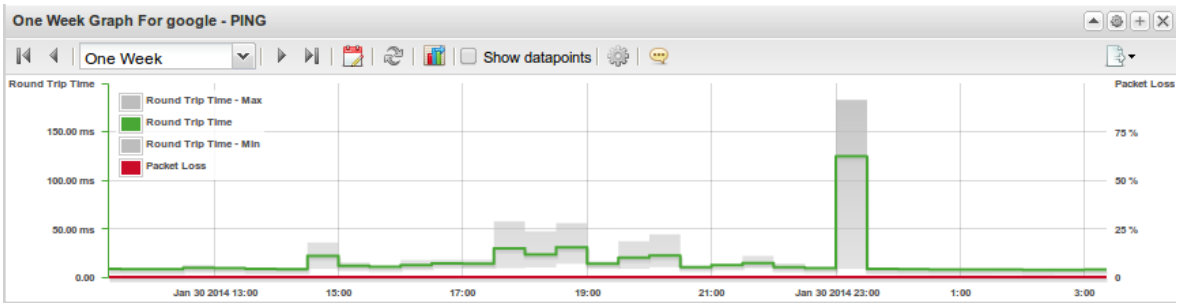
Monitorowanie popularnego i ogólnodostępnego serwera *google.com* pozwoliło na przetestowanie systemu z realnym systemem o dużym i bardzo zmiennym obciążeniu. Rezultaty monitorowania serwisu zostały przedstawione w formie wykresu na rys. 7.6. Nawet pobieżna analiza takiego wykresu pozwala zauważyć obecne w nim trendy. Na podstawie wykresu miesięcznego można zauważyć cykliczne wzrosty wartości maksymalnej dla zadanych przedziałów agregacji. Wykres ten pokazuje, iż występują one praktycznie codziennie. Analiza wykresu tygodniowego pozwala dodatkowo na określenie, że wspomniane skoki wartości maksymalnej występują w godzinach wieczornych. Ponadto łatwo zauważyć, że każdego dnia w godzinach popołudniowych i wieczornych występuje znaczny wzrost średniego czasu odpowiedzi serwera. Dzięki dynamicznemu charakterowi wykresów programu inGraph możliwe jest dokonanie dokładnej analizy widocznych na wykresie tendencji.

Rysunek 7.5. Wykres wartości czasu podróży pakietu oraz stopnia utraconych pakietów dla wl-500gPv2 w okresie, gdzie wystąpił wzrost stopnia utraconych pakietów.

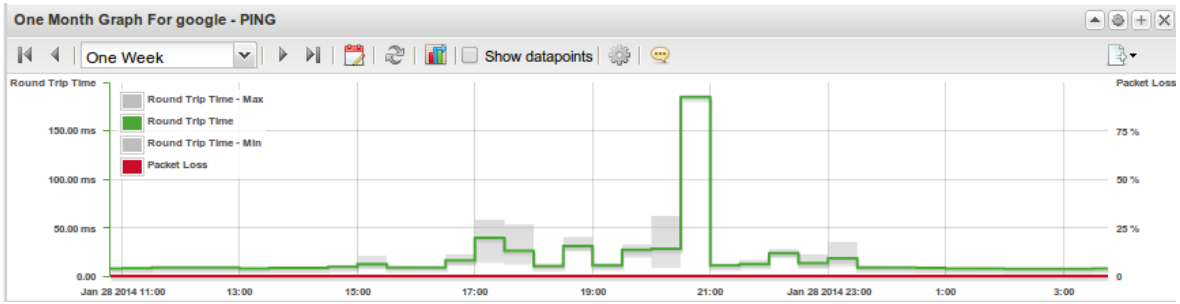


Rysunki 7.7, 7.8 oraz 7.9 przedstawiają dokładniejsze dane z przedziałów, w których notowano wzrosty czasu odpowiedzi. Analiza tych wykresów pozwoliła na potwierdzenie tezy o powtarzającej się tendencji. Łatwo można zauważyć, że w każdym z pokazanych przedziałów notowany jest stopniowy wzrost czasu odpowiedzi serwera od godziny 15. Godziny wieczorne to na każdym z wykresów zdecydowany wzrost czasu odpowiedzi, a także jego chwilowe skoki. Od około godziny 12 w nocy pojawia się spadek czasu odpowiedzi serwera, który utrzymuje się na niskim poziomie aż do godziny 15. Uzyskane rezultaty są zgodne z oczekiwanymi. Okres, w którym czas odpowiedzi serwera znacząco wzrasta, przypada na tak zwane internetowe godziny szczytu, czyli okres pomiędzy godziną 19 i 21. Zjawisko to zostało opisane w [4].

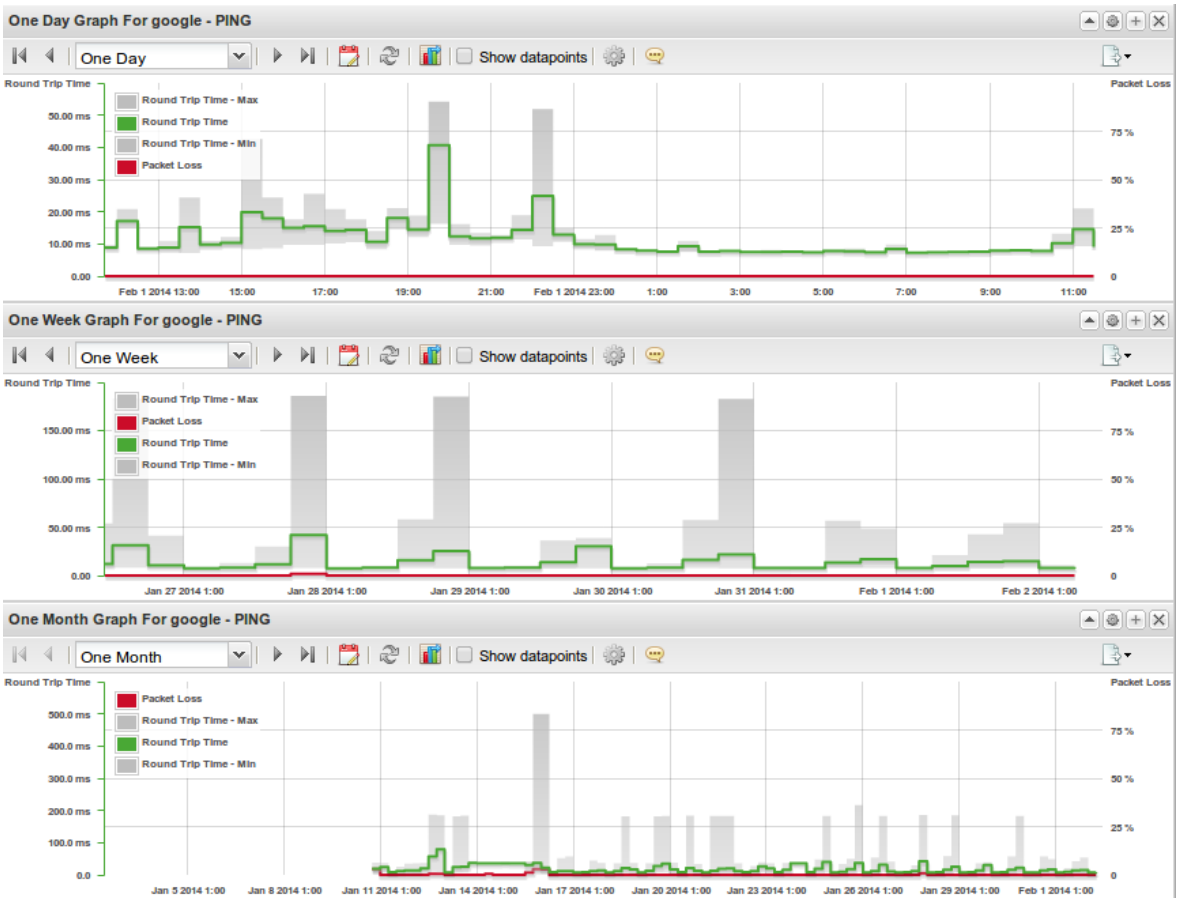
Rysunek 7.7. Wykres wartości czasu podróży pakietu oraz stopnia utraconych pakietów dla *google.com* dnia 30.01.2014.



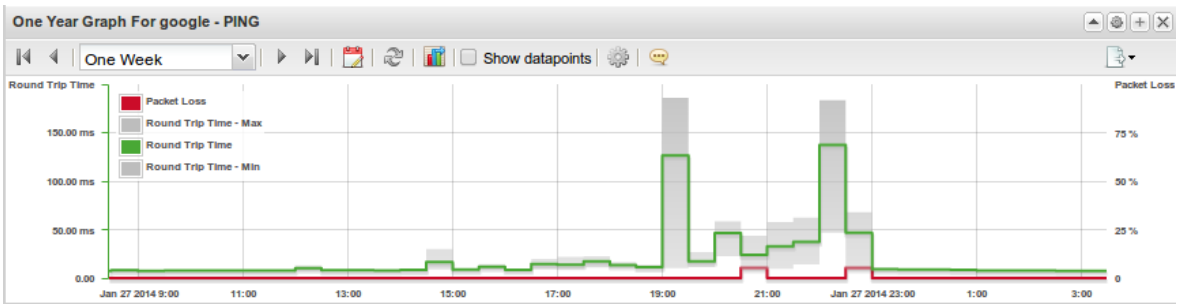
Rysunek 7.8. Wykres wartości czasu podróży pakietu oraz stopnia utraconych pakietów dla *google.com* dnia 28.01.2014.



Rysunek 7.6. Wykres wartości czasu podróży pakietu oraz stopnia utraconych pakietów dla google.com. Wykresy przedstawiają odpowiednio okres jednego dnia, tygodnia oraz miesiąca.



Rysunek 7.9. Wykres wartości czasu podróży pakietu oraz stopnia utraconych pakietów dla *google.com* dnia 27.01.2014.



7.4. Rezultaty dla klientów mobilnych

Monitorowanie klienta mobilnego odbyło się w czasie jednego tygodnia. Okres ten jest już znacznej długości, a codzienne używanie monitorowanych telefonów pozwoliło na wiarygodną symulację prawdziwych warunków funkcjonowania systemu.

Przeprowadzone testy pozwoliły wykazać poprawność działania wszystkich pożądanych mechanizmów. Pierwszym z przetestowanych mechanizmów było zapewnienie monitorowania stanu bieżącego usług i parametrów urządzeń mobilnych w sposób analogiczny jak urządzeń statycznych. System działał zgodnie z oczekiwaniami, dzięki czemu wszystkie ostrzeżenia były odpowiednio generowane. Przykładem takiego komunikatu jest powiadomienie administratora o stanie krytycznym baterii, które zostało przedstawione na rys. 7.10.

Rysunek 7.10. Pulpit stanów poszczególnych usług i parametrów klientów mobilnych.

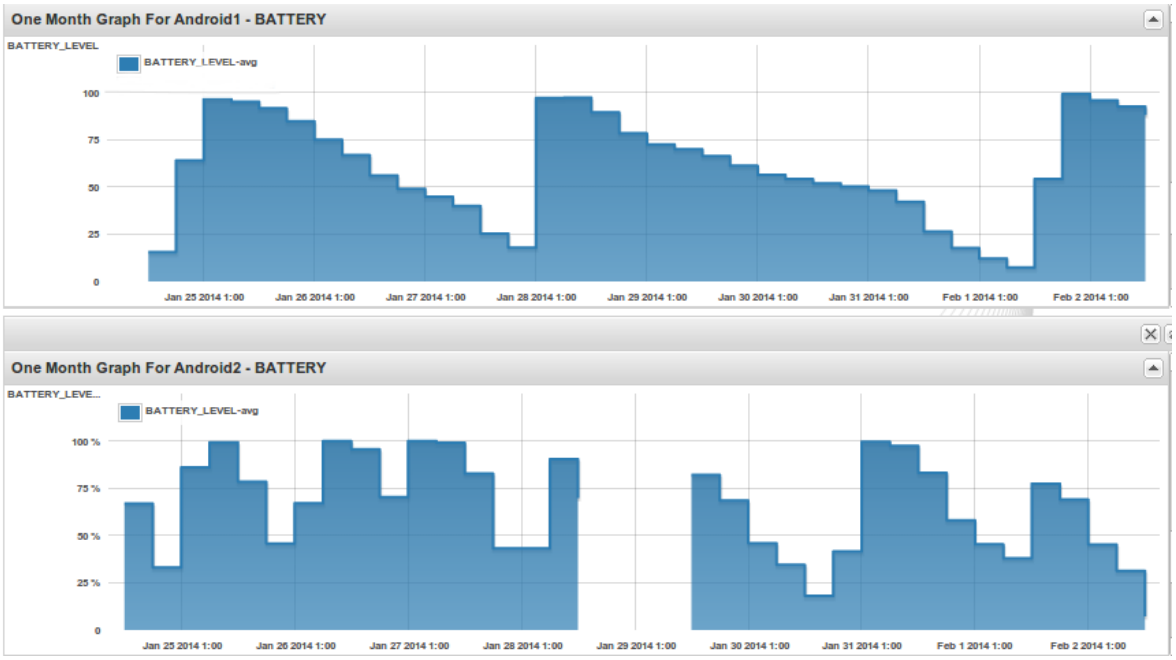
Service	Status	Last check	Duration	Info	Output
Host: Android1 (4 Items)					
APPLICATION	OK	2014-02-02 14:33:19	1w 3d 16h 14m 35s		APPLICATION=92
BATTERY	OK	2014-02-02 14:33:18	1d 30m 37s		BATTERY=87
CPU	OK	2014-02-02 14:33:19	3d 10h 56m 45s		CPU=8.275862
SIGNAL	OK	2014-02-02 14:37:18	1w 1d 15h 29m 42s		SIGNAL=53
Host: Android2 (4 Items)					
APPLICATION	OK	2014-02-02 15:10:40	20h 23m 14s		APPLICATION=56
BATTERY	CRITICAL	2014-02-02 15:10:39	1h 53m 15s		BATTERY=4
CPU	OK	2014-02-02 15:10:41	6d 19h 22m 23s		CPU=48.0
SIGNAL	OK	2014-02-02 15:10:42	1w 1d 23h 16m 46s		SIGNAL=46

Kolejną istotną funkcjonalnością, która była w tym czasie testowana, jest gromadzenie danych historycznych pochodzących od klienta mobilnego. Mechanizm gromadzenia tych danych i ich analizy działały podczas testów bez zarzutu. Pozwoliło to na wykonanie wykresów mierzonych wartości. Przykładowy wykres stanu baterii dla obu urządzeń wygenerowany na podstawie danych zebranych w czasie testów zawarto na rysunku 7.11.

Widoczna na jednym z wykresów przerwa wynika z braku danych w tym okresie. Ze względu na równolegle powstawanie pracy inżynierskiej Pana Marcina Kubika, który jest właścicielem tego urządzenia, konieczne było wyłączenie go na jeden dzień. Na przedstawionych wykresach można wyróżnić dwie fazy. Pierwsza z nich to rozładowywanie baterii. Okres ten można rozpoznać po zmniejszającym się pomiędzy kolejnymi pomiarami stanem baterii. Okres ładowania widoczny jest na wykresie jako czas gwałtownego wzrostu stanu baterii. Na podstawie wykresu można oszacować, iż bateria urządzenia Android1 była ładowana 3 razy, natomiast urządzenia Android2 co najmniej 6 razy. Wskazuje to na znaczne zużycie baterii w drugim urządzeniu lub jego bardzo intensywne użytkowanie.

Ponadto na podstawie przedstawionego wykresu można wnioskować o stylu użytkowania obu telefonów. Urządzenie Android1 ładowane jest do bardzo wysokiego poziomu baterii, a następnie użytkownik oczekuje z jego ładowaniem aż poziom baterii spadnie poniżej 20%. Użytkownik urządzenia Android2 natomiast bardzo często wykonuje ładowanie swojego telefonu nawet przy stanie powyżej 50%. W zależności od typu baterii znajdującej się w urządzeniu, takie działanie

Rysunek 7.11. Wykres stanu baterii urządzeń mobilnych w okresie testowania systemu. Pierwszy wykres (u góry) pokazuje stan baterii urządzenia Android 1, drugi (na dole) — urządzenia Android2.

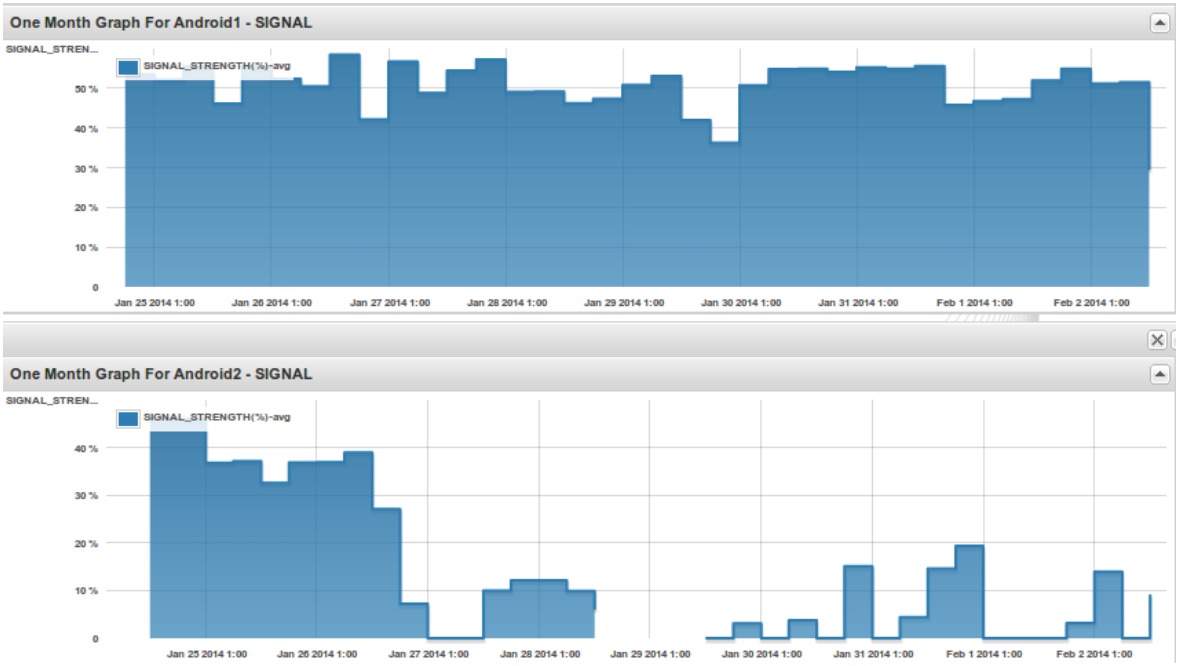


może skracać jej żywotność. Dzięki długoterminowemu monitorowaniu tego typu zachowań możliwe są badania wpływu profilu wykorzystania urządzenia na jego żywotność i awaryjność. Możliwość analizy sposobu użytkowania danego urządzenia może być dla Administratora bardzo pomocne. Może to zostać wykorzystane np. do lepszego zarządzania rozdziałem urządzeń mobilnych w firmie. Ponadto jeśli monitorowane urządzenia mobilne posiadają znaczną wartość, to śledzenie stylu ich użytkowania również jest niezwykle istotne, aby zapewnić ich długą żywotność.

W trakcie testów monitorowano również siłę sygnały Wi-Fi. Rezultaty pomiarów przedstawiono na rysunku 7.12. Ponownie na wykresie dla urządzenia drugiego widoczna jest przerwa wynikająca z chwilowego wyłączenia urządzenia. Na podstawie tych wykresów łatwo można zauważyć, że urządzenie Android1 posiada znacznie częstszy i znacznie lepszy dostęp do sieci Wi-Fi niż urządzenie Android2. Można wręcz zauważyć, iż urządzenie Android2 jest typowym klientem mobilnym, który dostęp do sieci uzyskuje bardzo sporadycznie. Słaba siła sygnału sieci Wi-Fi powoduje oczywiście zwiększone zużycie baterii, co jest widoczne na wykresie z rysunku 7.11.

W zastosowaniu produkcyjnym analiza takich danych jak dostępność sieci Wi-Fi w miejscu codziennego użytku urządzenia mobilnego jest niezwykle istotna. Firmy wydają bardzo duże pieniądze na zakup pakietów danych u operatorów sieci komórkowych w celu zapewnienia swoim pracownikom łączności z Internetem. Na podstawie analizy siły sygnału sieci dla każdego klienta można w dużo lepszy sposób zarządzać limitami danych otrzymanymi od operatora. Jeśli część klientów mobilnych przebywa przez większość czasu w zasięgu sieci Wi-Fi, to być może należy ograniczyć przyznane dla nich limity na rzecz zwiększenia przepustowości sieci

Rysunek 7.12. Wykres sygnału sieci Wi-Fi w okresie testowania systemu. Pierwszy wykres (u góry) pokazuje sygnał dla urządzenia Android1, drugi (na dole) — urządzenia Android2.



bezprzewodowej. Ponadto dane o sile sygnałów pochodzące z wielu urządzeń pozwalają na badanie wpływu rozmieszczenia punktów dostępowych na zasięg sieci w siedzibie firmy.

8. Podsumowanie

W ramach tej pracy wykonano projekt oraz implementację rozszerzenia systemu Icinga na potrzeby monitorowania klientów mobilnych. Zaprojektowany system został oparty na istniejących już elementach, których wykorzystanie pozwoliło na uzyskanie bardzo rozbudowanej funkcjonalności stosunkowo ograniczając nakład niezbędnej pracy.

Przed wykonaniem projektu przeprowadzono analizę dostępnych na rynku systemów monitorujących. Zidentyfikowane problemy monitorowania klientów statycznych oraz szczególny charakter klientów mobilnych wskazały na potrzebę opracowania nowych rozwiązań. W ramach analizy przedstawiono podstawowe możliwości otwartoźródłowych systemów *Cacti*, *Nagios* oraz *Icinga*. Porównanie tych systemów wykazało, że najlepszym systemem do rozbudowy w ramach tej pracy będzie system *Icinga*. Przedstawiono wymagania, jakie powinny zostać spełnione przez projektowany system monitorujący, aby umożliwiał on efektywne monitorowanie systemów mobilnych.

Kolejnym etapem przygotowania projektu była dokładna analiza możliwości systemu *Icinga* w kontekście zdefiniowanych wymagań. W ramach tej analizy przedstawiono ogólną architekturę systemu oraz zestaw dopuszczalnych jego konfiguracji. Dokonano również analizy dostępnych dodatków w kontekście ich funkcjonalności i jakości ich wykonania. Ze względu na brak gotowych rozwiązań pozwalających na monitorowanie klienta mobilnego zdefiniowano własny bezpieczny protokół komunikacyjny. Zapewnia on bezpieczny transport danych pomiędzy urządzeniem mobilnym a serwerem monitorującym. Konieczne było również wykonanie odpowiedniego dodatku do systemu *Icinga*, który umożliwiłby odbiór danych od klientów mobilnych i przekazanie ich do miejsc zgodnych z polityką całego systemu monitorującego.

Na podstawie projektu wykonano w języku C++ implementację dodatku NSCAv2. W trakcie prac wykorzystany został szkielet aplikacji *Qt*, a także biblioteka *boost*. Wszystkie algorytmy kryptograficzne wymagane do implementacji protokółu komunikacyjnego zostały zaimplementowane z użyciem biblioteki *Crypto++*.

Końcowym etapem tej pracy było wdrożenie przykładowej konfiguracji systemu *Icinga* zgodnie z przedstawionym projektem i z użyciem zaimplementowanego dodatku. Środowisko testowe zostało wykonane w domowej sieci lokalnej, jednak monitorowaniu podlegały również publiczne serwery *google.com* oraz *mion.elka.pw.edu.pl*. Dzięki życzliwości Pana Marcina Kubika, który wykonał implementację aplikacji mobilnej przeznaczonej na platformę Android, możliwe było włączenie do środowiska testowego również dwóch urządzeń pracujących pod kontrolą tego systemu.

Instancja systemu *Icinga* funkcjonowała nieprzerwanie przez trzy tygodnie, co zapewniło zgromadzenie reprezentatywnego zbioru danych. Wykorzystanie zewnętrznej infrastruktury zarówno do monitorowania publicznych serwerów, jak

i przekazywania danych od klientów mobilnych pozwoliło na symulację rzeczywistych warunków pracy takiego systemu.

Zebrane dane potwierdziły przydatność zaprojektowanego systemu. Na ich podstawie wskazano istotne trendy występujące w sieciach publicznych. Przeprowadzone testy wykazały również przydatność systemu monitorującego klienta mobilnego. Dane zebrane w trakcie monitorowania takiego urządzenia mogą posłużyć dużym firmom do redukcji kosztów utrzymania tych urządzeń i optymalizacji posiadanej infrastruktury sieciowej.

Jest bardzo wiele możliwych dróg rozwoju wykonanego projektu. Pierwszą z nich jest wykonanie narzędzia umożliwiającego w łatwy i intuicyjny sposób zarządzanie konfiguracją systemu monitorującego i jego dodatków. Drugą, zdecydowanie ważniejszą i dającą większe możliwości rozwoju systemu, jest wykonanie systemu eksperckiego. System taki mógłby na podstawie zgromadzonych danych historycznych (awariach, parametrach wydajnościowych, zdarzeniach itp.) wykrywać z wyprzedzeniem i ostrzegać o możliwości zaistnienia określonych sytuacji. Pozwoliłoby to administratorowi podejmować odpowiednie kroki zaradcze.

Bibliografia

[1] Agavi documentation. http://www.agavi.org/documentation/tutorial.

[2] Cacti project homepage. http://www.cacti.net/.

[3] CGI: Common Gateway Interface. http://www.w3.org/CGI/.

[4] Internet Rush Hour. http://en.wikipedia.org/wiki/Internet_Rush_Hour.

[5] Introducing JSON. http://www.json.org/.

[6] MCrypt project homepage. http://mcrypt.sourceforge.net/.

[7] Nagios Plugins Documentation. http://nagios-plugins.org/documentation/.

[8] Nagios project homepage. http://www.nagios.org/.

[9] Ceres project. https://github.com/graphite-project/ceres.

[10] Icinga Version 1.9 Documentation. http://docs.icinga.org/1.9/Icinga_v19_en.pdf.

[11] JasperReports Server. http://community.jaspersoft.com/project/jasperreports-server.

[12] Nagios Plugin Development Guidelines. https://nagios-plugins.org/doc/guidelines.html.

[13] Nagios Remote Plugin Executor. http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf.

[14] RRDtool Documentation. http://oss.oetiker.ch/rrdtool/doc/index.en.html.

[15] SNMP Technical Articles - overview. http://www.snmplink.org/articles/overview/.

[16] Whisper project. https://github.com/graphite-project/whisper.

[17] XML-RPC Specification. http://xmlrpc.scripting.com/spec.html.

[18] GNU General Public License version 2, 1991. http://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

[19] N. Neufeld C. Haen, E. Bonaccorsi. Distributed Monitoring System Based on Icinga. *International Conference on Accelerator and Large Experimental Physics Control Systems*, wolumen 13. CERN Geneva, 2011. http://accelconf.web.cern.ch/accelconf/icalepcs2011/papers/wepmu035.pdf.

[20] Ralph Johnson John M. Vlissides Erich Gamma, Richard Helm. *Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku.* Wydawnictwo Helion, 2010.

[21] Marcin Karbowski. *Podstawy kryptografii.* Wydawnictwo Helion, 2008.

[22] Marcin Kubik. Rozproszony monitoring systemów komputerowych, 2014.