Opracowanie skalowalnej platformy zdalnego zarządzania aplikacjami w systemach GNU/LINUX

Przebieg prac w III wariancie realizacji projektu z podziałem na poziomy gotowości

W poniższym raporcie opisano zakres i przebieg prac zrealizowanych w ramach projektu badawczo-rozwojowego pt. „Opracowanie skalowalnej platformy zdalnego zarządzania aplikacjami w systemach GNU/LINUX”. Końcowym efektem przeprowadzonych badań przemysłowych oraz prac rozwojowych jest udostępnienie zestawu produktów o cechach funkcjonalnych zgodnych z założeniami zdefiniowanymi w opisie innowacji produktowej i procesowej. Wypracowane metody wykraczają poza dziedzinę Digital Signage i mogą stanowić istotną część produktów związanych z innymi dziedzinami, jednakże przyjęte miary skuteczności dla badanych rozwiązań ukierunkowano na zastosowanie w tejże branży

Poziom III

Na etapie III poziomu realizacji projektu przeprowadzono badania w kierunku opracowania metod automatycznego reagowania i identyfikacji zdarzeń warstwy wizualnej zdalnych procesów w systemach GNU/Linux. Nadrzędnym celem tego etapu prac było osiągnięcie rezultatów o cechach innowacji produktowej w postaci udostępnienia rozwiązania wykraczającego funkcjonalnością poza produkty oferowane przez konkurencję, a także innowacji procesowej, umożliwiającej usunięcie określonych ograniczeń z dotychczasowego procesu wytwórczego i w konsekwencji dostosowanie modelu biznesowego. W części funkcjonalnej produktu, jako cel obrano udostępnienie możliwości otwarcia urządzeń docelowych (w systemach Digital Signage) na równoległe uruchamianie wielu niezależnych procesów w sposób zdalny, z możliwością uprzedniego, precyzyjnego zdefiniowania sposobu zachowania ich warstwy wizualnej. Dla porównania, istniejące produkty pozwalają na uruchamianie pojedynczych procesów w trybie pełnoekranowym lub umożliwiają sterowanie modułami aplikacji nadrzędnej, które imitują niezależne aplikacje. Kluczowy wpływ na część procesową ma cel uzyskania odpowiedniego poziomu niezależności technologicznej procesów

uruchamianych na urządzeniach docelowych. Przez usunięcie wymogów technologicznych, takich jak określony język programowania, ale także brak konieczności dostosowania aplikacji do zdefiniowanego API (interfejsu programistycznego), możliwe staje się wdrożenie szybkiego modelu outsource'ingowego lub inkorporowanie istniejących aplikacji na etapie przygotowania nowego projektu typu Digital Signage lub jego wdrożenia.

Etap analizy możliwości wykorzystania komunikacji międzyprocesowej w zadaniu śledzenia zmian w warstwie wizualnej zrealizowano w postaci czterech kolejno prowadzonych działań. Na działania te składa się analiza możliwości pozyskania niezbędnych informacji z autorskich aplikacji; analiza możliwości modyfikacji zewnętrznych aplikacji o otwartym kodzie źródłowym, których zbiór spełniałby funkcjonalne założenia projektu; analiza możliwości wykorzystania oraz ewentualnej modyfikacji warstwy menedżera okien serwera X11 (X Window System); określenie założeń wstępnych oraz estymacja kosztów opracowania i implementacji autorskiego rozwiązania działającego w warstwie menedżera okien X11 lub kompozytora Wayland. W wyniku prowadzonych działań zaklasyfikowano możliwość implementacji zestawu autorskich aplikacji jako działanie trywialne, lecz pozbawione cech innowacyjności oraz, przez silne nakierowanie na działania rutynowe i produkcyjne, stojące w sprzeczności z częścią biznesowych założeń projektu. Pomimo możliwości doboru zestawu otwartoźródłowych aplikacji, wypełniających obszar podstawowych funkcjonalności systemu Digital Signage, zidentyfikowano trudności w zakresie automatyzacji procesu modyfikacji tego rodzaju oprogramowania w związku z potrzebą unifikacji sposobu śledzenia zdarzeń poprzez komunikację międzyprocesową. W toku pozostałych działań, na podstawie sporządzonych wymagań wyodrębniono rozwiązania działające w warstwie menedżera okien serwera X11. Rozwiązania te, wykorzystujące w celu komunikacji międzyprocesowej głównie gniazda systemowe (unix sockets), zostały poddane testom oraz dalszym analizom pod kątem wykorzystania w pozostałych zadaniach. Warunkiem koniecznym z punktu widzenia opracowywanego rozwiązania była możliwość wirtualizacji wielu powierzchni renderowania (ekranów w pamięci operacyjnej), traktowanych przez serwer wyświetlania lub menedżera okien

jako fizyczny monitor, z jednoczesną możliwością mapowania współrzędnych na rzeczywisty ekran. Cecha ta pozwala osiągnąć efekt jednoczesnego uruchamiania wielu aplikacji działających w trybie pełnoekranowym przy zachowaniu ich pełnej dostępności. W rezultacie analiz dokonano definicji wymagań, zaprojektowano oraz zrealizowano rozszerzenie managera okien „Binary Space Partitioning Window Manager" dla serwera X11, umożliwiające przeprowadzenie dalszych prac i przyszłe wykorzystanie przemysłowe.

W toku analiz możliwości wykorzystania komunikacji międzyprocesowej do śledzenia zdarzeń w warstwie wizualnej wyodrębniono rozwiązania pozwalające na modyfikację atrybutów okien z wykorzystaniem gniazd systemowych. Z punktu widzenia opracowywanego systemu, zarządzającego w sposób dynamiczny przede wszystkim procesami, konieczne jest powiązanie zdarzenia z procesem przyczynowym, ze szczególnym uwzględnieniem możliwości wystąpienia pośredniego łańcucha procesów potomnych (ang. fork). Jako możliwą drogę identyfikacji kolejnych wiązań zbadano możliwość oparcia rozwiązania o drzewo procesów potomnych w wirtualnym systemie plików, stanowiącym interfejs jądra systemu operacyjnego oraz o atrybuty okien w serwerze X11. Jako podstawę tej części badań przyjęto zbiór 50 aplikacji podzielonych na kategorie według potencjalnego zastosowania w projektach typu Digital Signage, w tym: prezentacje multimedialne, odtwarzacze wideo, przeglądarki (galerie) zdjęć, przeglądarki dokumentów, przeglądarki internetowe, odtwarzacze audio, aplikacje informacyjne (np. czytniki RSS, prognozy pogody, kursy walut) oraz gry. Jako miarę skuteczności w tym przypadku przyjęto stosunek aplikacji spełniających kryteria kontroli (o możliwym do zidentyfikowania procesie przyczynowym) do mocy zbioru. Proces przetwarzania drzewa procesów uwarunkowano informacją z atrybutu okna X11, którego uzupełeninie nie leży jednak w zakresie formalnych wymagań protokołu. W wyniku otrzymano skuteczność na poziomie 96%, co uznano za wynik niewystarczający. W ramach prowadzonych prac zaproponowano metodę konstruowania reguł interpretowanych jako automatyczna reakcja (modyfikacja wskazanych atrybutów) na zdarzenie powiązane z procesem przyczynowym oraz rozszerzono zakres atrybutów poddanych analizie. Skuteczność na poziomie 100% uzyskano dzięki

możliwości skonstruowania reguł dedykowanych zarządzanym aplikacjom, z zachowaniem braku potrzeby ich modyfikacji. Działanie opracowanej metody zostało potwierdzone w przypadku autorskich aplikacji wewnętrznych oraz aplikacji zewnętrznych, wykorzystujących m.in. narzędzia takie jak *Qt 4.x*, *Qt 5.x*, *GTK+*, *Java Swing* oraz *Electron*. W trakcie ww. działań wyłoniono procesy, których wewnętrzna implementacja posiada mechanizm blokujący możliwość uruchamiania wielu instancji lub modyfikuje zachowanie instancji już istniejącej. Fakt ten doprowadził do konieczności opracowania dodatkowej metody separowania procesów oraz danych podręcznych aplikacji w czasie ich jednoczesnego uruchamiania.

W kolejnych działaniach podjęto realizację zadania zdalnego uruchamiania i monitorowania procesów systemowych oraz zaprojektowano i dokonano implementacji zestawu modułów współpracujących z opracowanym kontrolerem (aplikacją sterującą). Prace te miały na celu przystosowanie projektu do przeprowadzenia kolejnych badań oraz umożliwienie praktycznej weryfikacji dotychczasowych oraz przyszłych rezultatów. Moduły podzielono wg odpowiedzialności w następujący sposób:

- moduły zawierające algorytmy realizujące zadania komunikacji, w tym przesyłanie komunikatów sterujących i monitorujących przez REST API oraz w modelu publish – subscribe, a także transfer szyfrowanych pakietów binarnych;
- moduł zarządzania procesami systemowymi, działający w oparciu o opracowany system rejestracji deskryptorów procesów objętych kontrolą (podejście to pozwala m.in. na zwiększenie bezpieczeństwa przez izolację procesów administracyjnych od procesów aplikacji użytkownika);
- moduł komunikacji z warstwą wizualną;
- moduły logiki, gromadzące algorytmy odpowiedzialne za przepływ sygnałów sterujących, interpretację reguł reakcji na zdarzenia czy uruchamianie, śledzenie i zamykanie procesów za pośrednictwem modułu zarządzania procesami.

Ponadto, opracowano metody kontroli warstwy wizualnej uruchomionych aplikacji w zadanej

topologii ekranu. W celu odizolowania segmentów przypisanych poszczególnym procesom oraz dla zunifikowania metod kontroli okien, wprowadzono algorytm dynamicznego zarządzania wirtualnymi monitorami, co zgodnie z oczekiwaniami umożliwiło pracę w trybie pełnoekranowym wielu aplikacji jednocześnie.

Poziom IV

Procedura zdalnego uruchamiania zestawów aplikacji w rzeczywistych projektach Digital Signage związana jest z potrzebą transferu treści (zazwyczaj plików multimedialnych) do urządzenia docelowego. W etapie IV oraz w późniejszych etapach projektu rozpoczęto równoległe prace mające na celu podjęcie próby rozwiązania problemu wielokrotnego transferu tych samych treści w kolejnych zestawach aplikacji przy ograniczonej przestrzeni dyskowej urządzenia docelowego. Za miarę skuteczności przyjęto wpływ rozwiązania na redukcję zapotrzebowania na transfer, tzn. obserwowano liczbę bajtów przesyłanych między repozytorium treści a urządzeniami docelowymi w zbiorze scenariuszy użytkowych. Tradycyjne podejście obejmuje wykorzystanie przestrzeni dyskowej urządzenia docelowego jako pamięci podręcznej oraz przesyłanie plików za pomocą określonego interfejsu, którego działanie usuwa lub redukuje zawartość zbioru lokalnego przed przyjęciem nowego zestawu danych. Zaproponowane rozwiązanie obejmowało opracowanie innowacyjnego systemu plików, niezależnego od interfejsu aplikacji czy sposobu transferu danych, a także umożliwiającego zarządzanie lokalnymi zasobami w sposób dopasowany do zastosowania. Według wstępnej architektury tego rozwiązania, system plików działający w przestrzeni użytkownika (FUSE) imituje system macierzysty urządzenia dodając pośrednią warstwę chroniącą przestrzeń dyskową przed przepełnieniem. W momencie żądania zapisu danych, których rozmiar po dodaniu do obecnie zajętej przestrzeni przekroczyłby zadany limit (np. podczas przesyłania nowych plików multimedialnych) uruchamiany jest wewnętrzny moduł ewaluacji zasobów. Wynikiem działania tego modułu jest ocena każdego z lokalnych plików, dokonana na podstawie danych wejściowych, takich jak rozmiar pliku, data utworzenia czy data ostatniego dostępu, ale także danych kolekcjonowanych przez sam system plików, m.in. liczba otwarć pliku (osobno do

zapisu i do odczytu) czy łączny czas użytkowania danego zasobu (sumaryczny czas między każdym otwarciem i zamknięciem). Przed zapisem nowych danych, system plików automatycznie usuwa zasoby o najniższych ocenach do momentu, kiedy zaspokojone zostanie zapotrzebowanie na nowe miejsce. Badania nakierowano przede wszystkim na znalezienie formuł matematycznych dla różnych implementacji modułu oceniającego (zwanych dalej ewaluatorami), takich że dla znanych scenariuszy użytkowych najwyższe oceny zostaną przypisane do zasobów, których pozostawienie w pamięci zapobiegnie konieczności ponownego transferu tych danych w przyszłości. Rezultaty badań przedstawiły szeroki zakres wyników, wśród których w zależności od zastosowania (scenariusza użytkowego) liczbę przesłanych bajtów zmniejszono o od 30 do 50%, obierając za punkt odniesienia mechanizm działania pamięci podręcznej usuwającej zawsze najstarszy plik (FIFO). Na potrzeby projektu założono możliwość doboru wzoru (lub utworzenie nowego) dla danego zastosowania. Np. dla wdrożenia w muzeum prezentującym materiały promocyjne wystawy stałej (rzadko aktualizowane filmy) oraz aktualności związane z bieżącymi wydarzeniami (mniejsze pliki aktualizowane częściej), wybrana zostanie formuła faworyzująca większe zasoby o regularnym dostępie. Implementacji systemu plików dokonano z myślą zarówno o urządzeniach wbudowanych jak i komputerach stacjonarnych oraz serwerach. W aktualnych granicach funkcjonalnych mieści się arbitralny dobór formuły na etapie konfiguracji systemu, jednak przewidywane jest kontynuowanie prac w kierunku opracowania metod automatycznego doboru lub tworzenia formuł oceniających w zależności od charakterystyki urządzenia, np. z wykorzystaniem algorytmów uczenia maszynowego. Potencjalne zastosowania tej części projektu już w obecnej formie wybiegają poza branżę Digital Signage i mogą być użyte np. w serwerach proxy lub jako komponent aplikacji przetwarzających dużą ilość tymczasowych danych z możliwością ponownego wykorzystania.

Poziom V

W ramach prac nad architekturą umożliwiającą tworzenie, zarządzanie i dystrybucję nowego oprogramowania dla urządzeń końcowych, przeprowadzono analizę korzyści i zagrożeń

wynikających z zastosowania tradycyjnych i powszechnie dostępnych rozwiązań. Główne metody poddane ocenie to metoda oparta o repozytorium pakietów z zachowaniem sieci zależności i dystrybucję pakietów wykorzystujących jedną przestrzeń dla bibliotek współdzielonych oraz metoda dystrybucji pakietów z powielaniem zależności. Zidentyfikowano szereg ograniczeń tych metod, w szczególności nadmiarowy stopień złożoności i awaryjności podejścia wykorzystującego oprogramowanie zarządzające rozproszonymi pakietami instalacyjnymi, jak i zbędną redundancję w przypadku dystrybucji obrazów z powielaniem zależności. Na podstawie analizy i obserwacji testowanych podejść, wyznaczono pożądane cechy nowej architektury, z których najważniejsze to:

- brak konieczności śledzenia zależności aplikacji na etapie aplikowania aktualizacji,
- możliwość szybkiego przywracania dowolnej poprzedniej wersji systemu, szybkiego - tj. niewymagającego wykonywania operacji usuwania, kopiowania lub przenoszenia plików,
- brak konieczności stosowania określonego formatu pakietów na etapie sporządzania aktualizacji, tj. możliwość instalacji nowego oprogramowania w systemie wzorcowym w dowolny sposób,
- usunięcie nadmiarowości w postaci powielania bibliotek współdzielonych dla każdej aplikacji,
- odporność procesu aktualizacji jak i zaktualizowanego systemu na awarie powstałe wskutek przerwania trwających operacji (aktualizacji, startu, zapisu) np. przez nagłe odcięcie zasilania.

Aby osiągnąć ww. cele, rozwiązanie nie ogranicza zakresu działań jedynie do opracowania nowych metod tworzenia i aplikowania aktualizacji, ale wyznacza nowy sposób uruchamiania systemu operacyjnego w celu osiągnięcia wymaganych założeń.

Główne założenia tej części projektu dotyczą możliwości zarządzania systemem w sposób znany m.in. z systemów kontroli wersji, technologii kontenerowych oraz nadzorców maszyn wirtualnych z jednoczesnym zachowaniem możliwości bezpośredniego wykorzystania

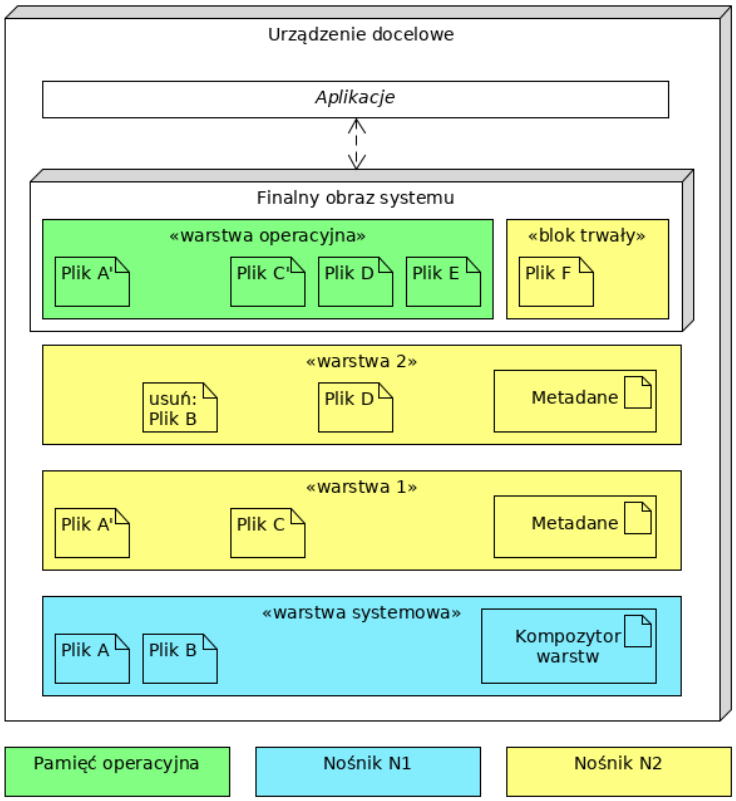
urządzeń peryferyjnych, interfejsów GPIO, akceleracji sprzętowej wideo (w tym 3D) oraz warstwy serwera okien bez opracowywania skomplikowanej warstwy pośredniczącej. Zachowanie wymienionych cech jest kluczowe w nowoczesnych projektach typu Digital Signage, gdyż coraz częściej urządzenia docelowe nie ograniczają się do wyświetlania treści z serwera, ale oferują użytkownikom interakcję oraz współpracują z różnego rodzaju urządzeniami peryferyjnymi. Poza spełnieniem kryteriów wyznaczonych po analizie wad dostępnych rozwiązań, jako kryterium oceny przyjęto także liczbę operacji zapisu i odczytu wykonywanych w czasie rozruchu i w czasie działania systemu operacyjnego. Wyznaczenie dodatkowego celu, jakim jest redukcja wykonywanych operacji (w szczególności zapisu), było spowodowane faktem istnienia problemu ograniczonej żywotności nośników fizycznych wykorzystywanych w urządzeniach docelowych, w szczególności pamięci typu FLASH w urządzeniach klasy embedded.

Etap zrealizowano poprzez modyfikację sekwencji rozruchowej urządzeń docelowych. Po uruchomieniu jądra systemu Linux przez firmware urządzenia lub program rozruchowy, następuje załadowanie wybranych, niezbędnych modułów jądra (m.in. sterowniki urządzeń pamięci masowej, obsługa wybranych systemów plików) i oddanie kontroli opracowanemu programowi komponującemu ostateczny kształt systemu przed jego właściwym uruchomieniem (program zwanym dalej kompozytorem warstw). Działanie kompozytora warstw polega na automatycznym doborze składników systemu docelowego wg wbudowanych reguł i metadanych dostarczanych wraz z kolejnymi aktualizacjami. Pracy kompozytora towarzyszą dwa dodatkowe założenia:

- pierwotna, dostarczona wraz z produktem wersja systemu operacyjnego stanowi dolną warstwę danych, która nigdy nie ulega bezpośredniej modyfikacji i traktowana jest jako stały zbiór zasobów tylko do odczytu,
- wszystkie zmiany w systemie wprowadzone po zakończeniu sekwencji uruchamiania, będą przechowywane w pamięci RAM jedynie jako różnica względem systemu podstawowego oraz nałożonych na ten system kolejnych "warstw" danych pochodzących z aktualizacji, będących wynikiem działania aplikacji użytkownika bądź samego systemu

operacyjnego.

Poza wskazanymi w konfiguracji wyjątkami (takimi jak rejestr zdarzeń systemowych czy pliki baz danych), operacje zapisu nie są fizycznie wykonywane na dysku twardym, co pozwala na uodpornienie systemu na awarie oraz jasne odseparowanie zmiennych elementów systemu od elementów stałych. Operacja sporządzenia nowej aktualizacji możliwa jest po uruchomieniu systemu wzorcowego (np. jako maszyny wirtualnej, na fizycznym urządzeniu symulującym urządzenie docelowe lub dowolnym rzeczywistym urządzeniu końcowym) z opracowaną sekwencją uruchamiania. Na potrzeby realizacji tego etapu zaprojektowano i zaimplementowano bibliotekę menedżera warstw, pozwalającą m.in. na rozpoczęcie rejestrowania zmian w górnej warstwie systemu (zapisane w pamięci RAM różnice między aktualnym stanem systemu a stanem najwyższej trwałej warstwy), wprowadzenie dowolnych zmian w systemie i eksport zmian do struktury danych, która ostatecznie stanowić będzie pojedynczy pakiet aktualizacyjny. Tak sporządzona struktura umożliwia jednorazowe zaaplikowanie wprowadzonych zmian na dowolnej liczbie urządzeń docelowych w postaci dodatkowej warstwy widzianej przez kompozytora warstw na etapie uruchamiania systemu. Poniżej przedstawiono poglądowy diagram ilustrujący sposób działania opracowanego rozwiązania.

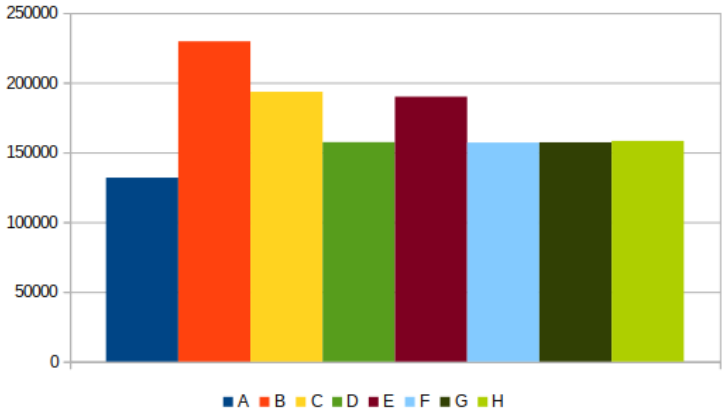


Przykład kompozycji warstw w zmodyfikowanej sekwencji uruchomieniowej systemu

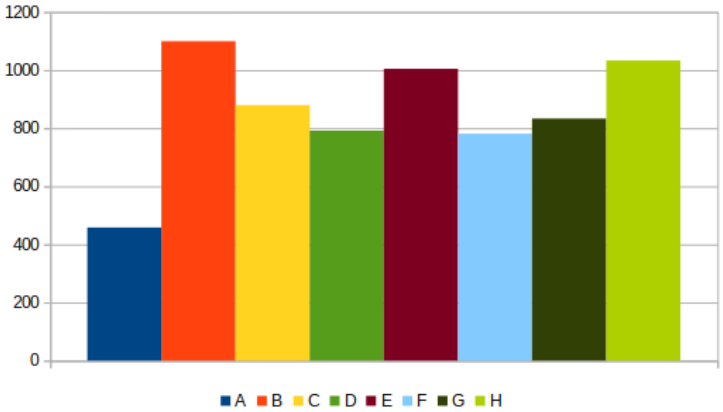
System operacyjny (procesy jądra) i aplikacje w przestrzeni użytkownika operują na plikach przedstawionych na diagramie jako “warstwa operacyjna” oraz na plikach pochodzących z bloku trwałego, w których zmiany zostaną zachowane dla kolejnego uruchomienia systemu. Podział ten nie wymusza zmiany lokalizacji zasobów. Przez tzw. wiązanie punktów montowania (ang. mount point binding) zachowują one swoją pierwotną lokalizację. W przytoczonym przykładzie w warstwie operacyjnej (najwyższej) widoczny jest plik A’ pochodzący z warstwy 1 (aktualizujący plik A z warstwy systemowej); plik C’, będący kopią pliku C zmodyfikowaną przez uruchomiony proces; niezmieniony plik D z warstwy 2; plik E utworzony wskutek działania aplikacji oraz modyfikowalny plik F, przechowywany na nośniku stałym. Dostęp do plików (odczyt) z niższych warstw poprzez warstwę operacyjną (jak w przypadku pliku D) nie powoduje alokacji przestrzeni w pamięci RAM. Dodatkowo, wyższe warstwy mogą zawierać metadane oznaczające dany zasób jako usunięty (plik B). W praktyce tego typu aktualizacja ma

znaczenie np. dla zbiorczych katalogów przechowujących automatycznie przetwarzane pliki konfiguracyjne.

Kompozytor warstw tworzy ostateczny, spójny obraz systemu przez nakładanie kolejnych warstw od najstarszej do najnowszej, zastępując przy tym starsze pliki nowszymi w przypadku powielenia zasobów. Praca kompozytora nie spowalnia w znaczący sposób sekwencji uruchamiania, gdyż na żadnym etapie składania warstw nie występuje operacja kopiowania ani przenoszenia plików - w tym celu wykorzystywane są jedynie operacje zarządzania punktami montowania zasobów i cechy wybranych systemów plików. Dodatkowo, poprzez przekierowanie operacji wejścia/wyjścia do najwyższej warstwy i przechowywanie rezultatów w pamięci operacyjnej, znacząco zredukowano liczbę operacji wykonywanych na fizycznym nośniku, tym samym zwiększając jego żywotność i redukując koszty utrzymania systemu. Wykresy poniżej przedstawiają liczbę operacji zarejestrowanych przez narzędzie „iostat" dla tych samych urządzeń (A – H) bez modyfikacji sekwencji uruchomieniowej (wykres 1) oraz z opracowaną sekwencją uruchomieniową (wykres 2). We wszystkich przypadkach operacje zapisu zdarzeń systemowych kierowane były na fizyczny nośnik (zachowanie operacji na plikach z katalogu *„/var/log")*. Wyniki wskazują, że już w przypadku urządzenia realizującego zadanie wyświetlania prostej prezentacji multimedialnej (A), rozmiar danych zapisanych na nośniku podczas sekwencji uruchomieniowej został zredukowany z 131833.8KB do 457.8KB. Największą oszczędność zaobserwowano w przypadku urządzeń z zestawami aplikacji operujących na tymczasowych danych generowanych na potrzeby bieżącej sesji (np. dane podręczne przeglądarek internetowych).



Wykres 1: liczba KB zapisanych podczas niezmodyfikowanej sekwencji uruchomieniowej



Wykres 2: liczba KB zapisanych podczas zmodyfikowanej sekwencji uruchomieniowej

Urządzenie testowe	Dane (KB) zapisane podczas niezmodyfikowanej sekwencji uruchomieniowej	Dane (KB) zapisane podczas zmodyfikowanej sekwencji uruchomieniowej
A	131833.8	457.8
B	229599.4	1099.4
C	193413.8	879.4
D	157345	791.4
E	189909.8	1004.2
F	157009	781
G	157154.6	833

H	158201	1033

Zestawienie wartości z wykresów 1 i 2

Każda nowa struktura aktualizacji zawiera zbiór metadanych, które informują kompozytor m.in. o wymaganych warstwach niższych. Po przeprowadzeniu poprawnej sekwencji uruchamiania, warstwy niewykorzystywane mogą zostać usunięte. Przywracanie poprzedniej wersji systemu nie wymaga dodatkowych operacji kopiowania, usuwania ani przenoszenia plików, gdyż jedyną zmianą jest wskazanie aktywnych warstw w konfiguracji kompozytora. Dodatkowo, biblioteka menedżera warstw implementuje funkcje ułatwiające administrowanie i optymalizację pracy kompozytora, w tym scalanie warstw, szyfrowanie, klonowanie czy manualne usuwanie warstw i warstw zależnych. Ograniczeniem opracowanej architektury może być zwiększone zużycie pamięci operacyjnej względem tradycyjnego podejścia, szczególnie w przypadku operacji zapisu wykonywanych na dużych plikach. W takich okolicznościach system przewiduje wykorzystanie pamięci trwałej jako miejsce przechowywania tymczasowych zmian (możliwie na oddzielnym nośniku, którego uszkodzenie nie spowoduje przerwy w świadczeniu wszystkich usług i nie uniemożliwi przeprowadzenia czynności diagnostycznych). Dalsze prace, wychodzące poza zakres bieżących wymagań, przewidują badania możliwości monitorowania operacji wejścia/wyjścia wykonywanych przez uruchamiane aplikacje i automatyczne zarządzanie przydzielaniem typu pamięci.

Czynności zmierzające do realizacji zadań związanych z przygotowaniem zestawów algorytmów do autoweryfikacji poprawności działania usług w systemie wraz z możliwością przywrócenia ostatniej poprawnej wersji systemu zostały podjęte na etapie prowadzenia prac związanych z modyfikacją sekwencji uruchomieniowej. Opracowana architektura zakłada autoweryfikację poprawności działania usług w systemie w trzech krokach, z których dwa pierwsze wykonywane są na wczesnym etapie uruchamiania systemu przez opisany powyżej kompozytor warstw. Z punktu widzenia podprogramu przygotowującego ostateczny system do późniejszej inicjalizacji, przyjmując numerację warstw od najstarszej do najnowszej,

wykorzystywana jest zawsze warstwa o numerze N-K, gdzie N to liczba wszystkich warstw z aktualizacjami, a K to liczba przeprowadzonych testów zakończonych wynikiem negatywnym. Aktywacja (montowanie) wskazanej warstwy poprzedzone jest odczytem metadanych z informacją o identyfikatorach warstw niższych. Po zamontowaniu tych warstw (wymaganych poprzednich aktualizacji, których nie zastępuje aktualizacja bieżąca) oraz warstwy docelowej, uruchamiany jest tzw. „test poczytalności", weryfikujący np. działanie wymaganych komponentów systemu, obecność niezbędnych zasobów, spójność danych potwierdzona sumami kontrolnymi oraz poprawne rozwiązanie zależności podstawowych aplikacji.

Po uzyskaniu pozytywnego wyniku testu poczytalności, uruchamiany jest program testujący dystrybuowany wraz z aktualizacją. Zadaniem tego etapu jest weryfikacja kompatybilności aktualizacji z systemem podstawowym oraz przeprowadzenie wszystkich dodatkowych asercji charakterystycznych dla bieżącej warstwy. Przebieg tego etapu zależy wyłącznie od dostawcy aktualizacji i w minimalnej postaci (jak np. aktualizacja plików graficznych) może zawsze zwracać wynik pozytywny w celu przyśpieszenia sekwencji startu, a weryfikację spójności danych zostawić mechanizmom dystrybucji opisanych w dalszej części dokumentu. W przypadku negatywnego wyniku testu poczytalności lub testu warstwy, parametr K podlega inkrementacji, wskutek czego kompozytor podejmuję próbę przygotowania systemu w postaci nieobejmującej ostatniej aktualizacji. W pesymistycznym przypadku (gdzie N = K) uruchomiony zostanie system podstawowy dystrybuowany wraz z produktem.

W trzecim kroku uruchamiany jest test aplikacji wykonywany po zakończeniu sekwencji startu. Na tym etapie możliwe jest zweryfikowanie poprawności działania tych elementów przestrzeni użytkownika, na które wpływa najnowsza aktualizacja np. rekonfiguracja sieci, zmiana parametrów wyświetlania czy poprawne uruchomienie wymaganych usług systemowych. W przypadku negatywnego wyniku testu na tym etapie, aktualna warstwa oznaczana jest jako wadliwa i następuje restart systemu, przez co kompozytor warstw w kolejnej sekwencji startowej rozpocznie weryfikację z pomniejszoną wartością parametru N. Weryfikację rezultatów powyższej części projektu sprowadzono do przeprowadzenia testów dla przypadków testowych (poprawnych i błędnych aktualizacji) i weryfikacji wyników. Administrator systemu może

niezależnie od wyników testów włączyć lub wyłączyć warstwę z użycia.

Poziomy VI - VIII

Zakładana we wczesnych etapach projektu konieczność rekonfiguracji programu rozruchowego związana była z założeniem przeprowadzania całości procedury aktualizacji w przestrzeni użytkownika, po pełnej inicjalizacji systemu. Z uwagi na opracowanie rozwiązania modyfikującego proces sekwencji startowej systemu i zmianę strategii aplikowania zmian w systemie, edycja konfiguracji programu rozruchowego przestała być elementem koniecznym. W szczególnych przypadkach podjęto decyzję o całkowitym wyłączeniu udziału programu rozruchowego z sekwencji startowej.

Zgodnie z opracowaną wcześniej architekturą sporządzania, wersjonowania i dystrybucji aktualizacji, pojedyncza aktualizacja przyjmuje formę warstwy aplikowanej przez kompozytor warstw podczas kolejnego rozruchu systemu. Każdy pakiet przed umieszczeniem w zdalnym repozytorium podlega szyfrowaniu (oraz opcjonalnie kompresji). Problem polegający na konieczności pobierania i przechowywania dużych warstw przed ich odszyfrowaniem na urządzeniu docelowym rozwiązano umożliwiając dostęp do zdalnego repozytorium przez sieciowy system plików. Dodatkową zaletą tego rozwiązania jest możliwość aktualizacji maszyn w trybie offline, bez dostępu do zdalnego repozytorium. Wówczas moduł odczytujący pakiety aktualizacyjne nie rozróżnia zdalnego źródła danych od lokalnego, np. w postaci nośnika USB.

Podjęcie próby zastosowania protokołu Wayland w opracowywanym systemie poprzedzono analizą korzyści wynikających z zastąpienia serwera okien X przez autorskie rozwiązanie implementujące interfejs zgodny z pozostałą częścią systemu. W przypadku urządzeń końcowych o architekturze x86 (np. komputery klasy mini PC), które realizują podstawowe założenia projektu z użyciem serwera X11, nie stwierdzono znaczącego zysku płynącego z powtórnej implementacji tych samych funkcjonalności z użyciem protokołu Wayland. Obecna architektura systemu wyklucza także użycie serwera pośredniczącego

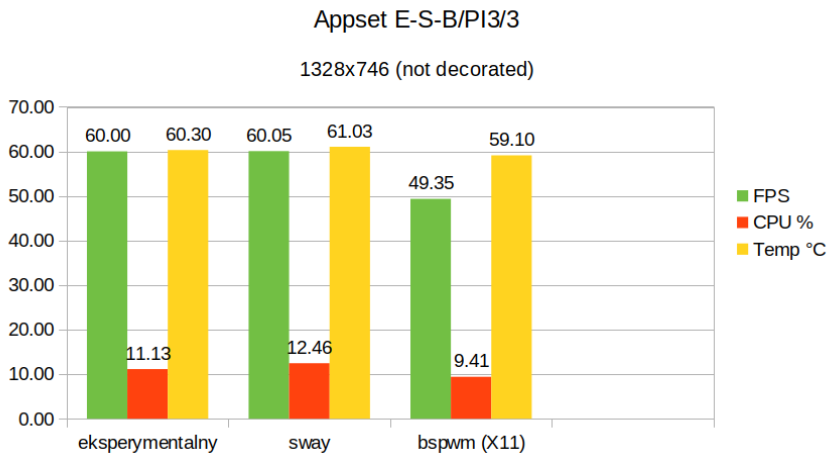
XWayland w celu kontynuowania wsparcia dla zewnętrznych aplikacji. Jednakże z uwagi na rozwój i wzrost dostępności urządzeń typu IoT oraz komputerów klasy SBC, których potencjał wydajnościowy (w szczególności wykorzystanie procesorów graficznych) nie jest w pełni dostępny przy użyciu X11, podjęto próbę implementacji podzbioru funkcjonalności opracowywanego systemu z wykorzystaniem protokołu Wayland. Główne cele na tym etapie prac zostały zdefiniowane następująco:

1. Opracowanie rozwiązań z zakresu wytwarzania oprogramowania zdolnego do działania zarówno w kompozytorze Wayland jak i X11 z ukierunkowaniem na rozwiązania niewymagające zmian w przypadku potrzeby skorzystania z alternatywnego protokołu. W tym zakresie rozpatrywano QPA (Qt Platform Abstraction), SDL 2.0 (Simple DirectMedia Layer) oraz EFL (Enlightenment Foundation Libraries). Po analizie wstępnych wyników, w dalszych pracach wykorzystano QPA.
2. Wytworzenie kompozytorów Wayland z wykorzystaniem modułu QtWaylandCompositor, bibliotek wlroots lub KWayland.

Poza weryfikacją możliwości funkcjonalnych i potencjalnej przewagi badanego rozwiązania w szczególnych przypadkach, poddano ocenie działanie wytworzonych aplikacji testowych pod kontrolą opracowanego eksperymentalnego kompozytora okien, istniejących implementacji kompozytorów Wayland ogólnego przeznaczenia, a także przy użyciu menedżera okien X11. Przykładowe wyniki pomiarów dla aplikacji prezentujących animację wygenerowaną przy użyciu silnika cząsteczkowego przedstawiono poniżej.

Wyniki dla zestawu “Appset E-S-B/PI3/3”, 1328x746, bez dekoracji okien:

Kompozytor	FPS	CPU %	Temp °C	Mem (KiB)
eksperymentalny	60.00	11.13	60.30	21120.00
sway	60.05	12.46	61.03	24037.33
bspwm (X11)	49.35	9.41	59.10	16424.00



Wstępne próby wykazały, iż wybrane funkcjonalności systemu w zakresie Digital Signage mogą zostać zrealizowane jako oprogramowanie wbudowane, dedykowane wybranym urządzeniom i aplikacjom. Kompletny projekt i implementacja tej części nie oferuje jednak wystarczającej przewagi nad X11 w ogólnym zastosowaniu, a działania te wykraczają poza harmonogram prac projektu i mogą zostać zrealizowane w kolejnych iteracjach cyklu rozwoju systemu.

Implementacja testowa

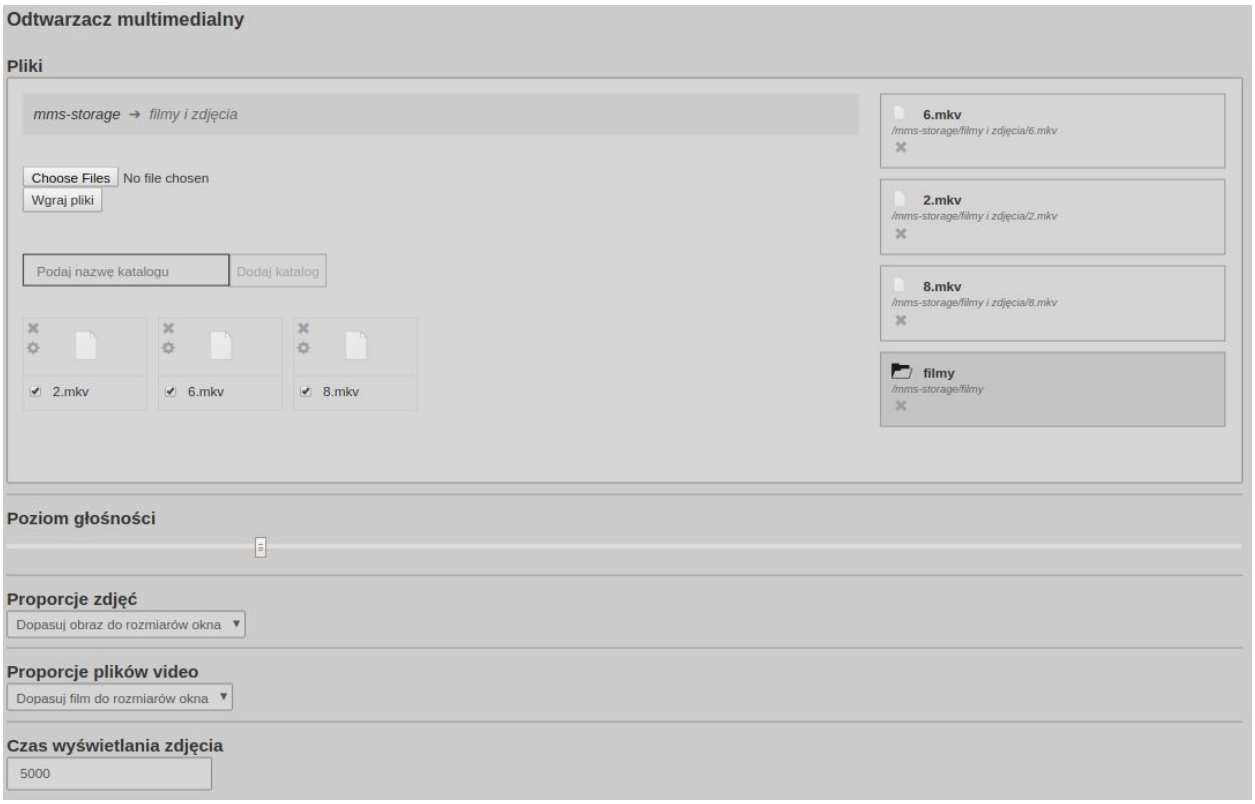
Na poszczególnych etapach podejmowano równoległe prace rozwojowe oraz rutynowe prace inżynierskie w celu wytworzenia oprogramowania dla urządzeń końcowych serwera głównego, aby umożliwić bieżącą, praktyczną weryfikację zakresu funkcjonalnego opracowywanego systemu. W kolejnych iteracjach cyklu rozwoju oprogramowania udostępniono szereg rozwiązań w formie zbliżonej w kluczowych aspektach do formy produktu końcowego. Dostęp do centralnego punktu sterującego zrealizowano w postaci aplikacji internetowej zgodnej z koncepcją SPA. W celu zapewnienia interoperacyjności nowego systemu z już posiadanymi systemami i komponentami narzędziowymi, wdrożono dwa niezależne modele komunikacji (request-response oraz publish-subscribe) ze zgodną, wewnętrzną specyfikacją opisującą zadania do wykonania przez układy docelowe. Pierwsze iteracje cyklu rozwoju zakładały wytworzenie komponentu, który udostępnia przestrzeń roboczą o wysokim

stopniu dowolności w zakresie tworzenia sekcji przewidzianych dla układu docelowego z przeznaczeniem dla administratora systemu. W trakcie prac zgodnych z takim podejściem, stopniowo zaczęła uwydatniać się nieefektywność rozwiązania (podatność na pozostawianie niewypełnionych przestrzeni, konieczność odpowiedniego doboru proporcji, rozmiarów bez przesłaniania wcześniej utworzonych sekcji). Po analizie wad pierwszej implementacji, podjęto decyzję o opracowaniu i wprowadzeniu rozwiązania opartego o strukturę drzewa binarnego, co umożliwi tworzenie sekcji ekranu poprzez podzielenie fragmentu wertykalnie lub horyzontalnie z możliwością dynamicznej zmiany proporcji wskazanego podziału i automatycznym uwzględnieniem w sekcjach podrzędnych. Wprowadzenie modyfikacji oraz zaaplikowanie ich w urządzeniu docelowym powoduje zmiany jedynie w sekcjach, w których wystąpiła intencja zmiany treści, a mechanizmy wykonają synchronizację danych do urządzeń docelowych w sposób różnicowy.

Przykładowe wyniki podziału ekranu na sekcje dla zdalnych aplikacji

Komponentem pośredniczącym w procesie zdalnej kontroli procesów jest moduł odpowiadający za translację wytworzonego szablonu ekranu na format zgodny z urządzeniami docelowymi. Działanie tego mechanizmu polega na przetworzeniu drzewa binarnego, przypisaniu wybranych procesów do poszczególnych jego węzłów, normalizacji wartości mapowanych na ekran

docelowy oraz powiązanych metadanych, niezbędnych do poprawnego działania systemu. Przypisane do sekcji aplikacje, wybrane z listy autoryzowanych programów (także pochodzących od zewnętrznych dostawców), są w kolejnym kroku konfigurowane przez dedykowane im formularze ustawień. Aktywacja tak utworzonego szablonu ekranu następuje po wskazaniu urządzenia lub grupy urządzeń i tylko po wcześniejszym uzyskaniu pozytywnego wyniku weryfikacji szablonu. Dalsze prace zakładały udostępnienie repozytorium danych przeznaczonego dla administratora systemu. Wytworzenie takiego autorskiego komponentu podyktowane było koniecznością wprowadzenia filtrów dla wskazanych aplikacji oraz możliwością wskazywania zasobów przeznaczonych do prezentacji na urządzeniach docelowych. Takie rozwiązanie pozwoliło też opracować moduły zdolne do współdzielenia treści (np. multimedialnych) przez różne szablony podziału ekranu oraz w sposób dowolny zmieniać kolejność prezentowanych zasobów. Mając na uwadze jednokierunkową synchronizację danych (z repozytorium danych do urządzenia docelowego) wprowadzono niepowiązane mechanizmy usuwania treści z repozytorium oraz wyłączania wskazanego zasobu dla każdego podziału ekranu. Do zrealizowanych prac uzupełniających należy dołączyć standardowe mechanizmy znane z dziedziny zarządzania plikami i katalogami.



Formularz konfiguracji wybranej aplikacji “Odtwarzacz multimedialny” na etapie sporządzania zestawu aplikacji w panelu administracyjnym

Podsumowanie

Po dokonaniu analiz i opracowaniu metod z poziomu III realizowanego projektu, wraz z równoległym rozpoczęciem prac związanych z etapem IV, podjęto testową implementację opisanych rozwiązań. Rozszerzenie warstwy managera okien serwera X11 pozwoliło na zbudowanie modułowego systemu zdalnego zarządzania procesami z warstwą wizualną o niedostępnym do tej pory w dziedzinie Digital Signage poziomie rozbudowy. Skalowalność technologiczną zapewnia brak wymagań co do stosowanej technologii dla aplikacji końcowych – istniejące odtwarzacze multimediów, aplikacje internetowe czy dedykowane aplikacje dziedzinowe mogą być uruchamiane w trybie „kiosk” jednocześnie na wielu wirtualnych monitorach w ramach jednego fizycznego ekranu. Towarzyszący częstym zmianom zestawów aplikacji problem transferu plików multimedialnych został rozwiązany przez projekt

inteligentnego systemu plików, przezroczystego dla mechanizmów zapisu i automatycznie podejmującego decyzje o zwolnieniu przestrzeni dyskowej.

W ramach realizacji wymagań dotyczących pozafunkcjonalnej części projektowanej platformy, opracowano alternatywną sekwencję uruchomieniową systemu GNU/Linux, otwierającą nowe możliwości dla bardziej wydajnych metod aktualizacji zdalnych urządzeń, a także redukujące koszt utrzymania wdrożonych systemów. Jednocześnie opracowano rozwiązania umożliwiające tworzenie i zarządzanie pakietami aktualizacyjnymi w sposób pozwalający na odizolowanie sposobu zapisu od rodzaju użytego nośnika, zachowując tym samym pełną przenośność metadanych. Algorytmy i narzędzia opracowane w tym poziomie stanowią samodzielną całość i mogą być stosowane nie tylko w obrębie platformy zdalnego zarządzania aplikacjami. Dzięki możliwości rozdzielenia warstw systemu i komponowania jego ostatecznego kształtu w sposób przypominający np. ten z systemów kontroli wersji lub nadzorców maszyn wirtualnych, ale z jednoczesnym i bezpośrednim dostępem do zasobów sprzętowych (GPIO, procesor graficzny, urządzenia peryferyjne) rozwiązanie to może być stosowane w szeroko pojętym IoT, jako podstawa w stanowiskach pracowniczych czy w dedykowanych aplikacjach przemysłowych.

W ostatnich etapach projektu dokonano udanej próby przeniesienia istniejących i funkcjonujących już produktów na opracowaną platformę. Niezależność od technologii i brak konieczności spełniania restrykcyjnych wymagań interfejsu programistycznego pozwoliły na przeprowadzenie migracji bez ingerencji w istniejące oprogramowanie. W kolejnych etapach rozwoju projektu przewidywane są dalsze badania nad możliwościami wykorzystania protokołu Wayland w systemach wbudowanych, optymalizacja istniejących algorytmów, projekt systemu plików wspomagający opracowany system kompozycji warstw oraz rozwój narzędzi administracyjnych.