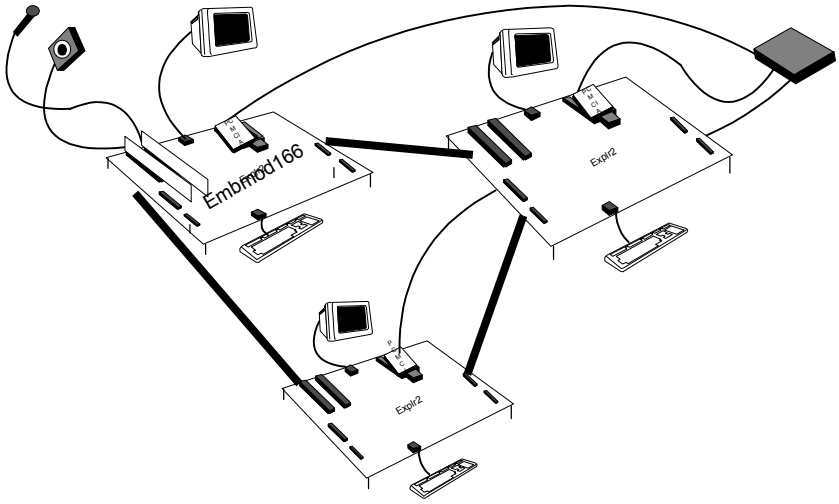
Adaptacyjny algorytm diagnozowania sieci komputerowej

Charakterystyka obiektu diagnozowania



Sieć komputerowa typu
punkt – punkt, (Ethernet)

System operacyjny węzła: QNX

Liczba węzłów: n≥3

Ogólne założenia dla algorytmu

- model diagnostyki PMC;
- węzły nieprawidłowo działające nie wykonują algorytmu (fail-silent fault) , lub też pakiety od nich są rozpoznawane i odrzucane;
- węzeł nie może ulec awarii i zostać naprawiony (zrestartowany) w przerwie miedzy dwoma testami (zmiana stanu musi zostać dostrzeżona);
- sieć jest reprezentowana grafem zwykłym bez pętli (miedzy sąsiadami istnieje komunikacja dwustronna);
- operacje takie jak konstrukcja i wysłanie pakietu oraz odebranie i obsługa pakietu są względem siebie atomowe (TRANSAKCJE NIEPODZIELNE) .

Opis zasady działania

- Adapt2 jest adaptacyjnym algorytmem służącym diagnozowaniu węzłów w sieciach i systemach wieloprocesorowych (wielokomputerowych). Algorytm dopuszcza dynamiczne uszkodzenia i naprawy węzłów i połączeń między węzłami w czasie jego działania.
- W wyniku jego działania konstruowany jest przydział testów, który gwarantuje, że każde uszkodzenie węzła zostanie wykryte przez co najmniej jeden sprawny węzeł sieci. Adapt2 wykonuje się w dwóch fazach: pasywnej i aktywnej.
- W fazie pasywnej węzeł okresowo testuje sąsiadów wg przydziału testów. Faza aktywna jest inicjowana w momencie wykrycia zmiany stanu węzła (uszkodzenie bądź restart po naprawie). Prawidłowo działające węzły konstruują wtedy nowy przydział testów i odświeżają swoją diagnozę wszystkich jednostek w sieci.
- **Pakiet nazywany jest kompletnym** kiedy przejdzie przez wszystkie sprawne węzły i powróci do pierwotnego nadawcy (zwanego *root*). Ścieżka przebyta przez pakiet determinuje przydział testów. Przydział testów ma strukturę drzewa. Każdy węzeł testuje węzeł od którego otrzymał dany pakiet (ojca) i wszystkie węzły do których go wysłał (dzieci).

Węzły do których pakiet nie dotarł są uznawane za błędne. Kompletny pakiet jest wysyłany powrotnie przez węzeł root tak, aby każdy węzeł miał całkowite informacje o stanie sieci i przydziale testów. Kiedy pakiet transmitowany z węzła n_j dochodzi do węzła n_i pierwszy raz, n_i zapisuje *pkt.from[i]=j*. Gdy pakiet przemierzy wszystkie wolne od uszkodzeń węzły, ścieżka propagacji jest zarejestrowana w *pkt.from*

Struktury danych

Węzły

Struktura danych posiadanych przez każdy węzeł

```
int nodeid                  //identyfikator węzła
int nodenum                 //liczba węzłów w sieci
int event[nodenum]          //tablica znaczników czasów każdego węzła - do ustalania
                            //porządku zdarzeń
flag diagnosis[nodenum]     //{uszkodzony, sprawny} - syndrom sieci
flag tests[nodenum]         //{uszkodzony, sprawny, nie testuj} - spodziewany wynik testu
```

- Każdy węzeł posiada unikalny identyfikator ***nodeid***, przechowuje liczbę wszystkich węzłów występujących w sieci ***nodenum*** jako wartość stałą. Algorytm może być modyfikowany do zastosowań w sieciach o różnych ilościach węzłów poprzez wprowadzenie stosownych struktur danych (np.: poprzez zastosowanie listy dynamicznej zamiast macierzy, oraz podejmowania odpowiednich akcji w przypadku otrzymania informacji dotyczących nowego węzła).
- Podstawową strukturą danych używaną przez algorytm jest tablica znaczników czasu zdarzeń ***event*** używana do porządkowania zdarzeń. Tablica znaczników czasu zawiera pozycje odpowiadające poszczególnym węzłom, jedna pozycja dla każdego węzła. Węzeł n_i zwiększa swój znacznik czasu, *event[i]* kiedykolwiek nowe zdarzenie stwierdzające uszkodzenie jest wykryte przez węzeł n_i.
- Każdy węzeł przechowuje tablicę ***diagnosis*** zawierającą flagi po jednej dla każdego węzła, identyfikujące węzeł jako uszkodzony ‘Faulty’ lub sprawny ‘FaultFree’. Tablica ***tests*** zawiera oczekiwane wyniki testów. Na przykład jeżeli n_i przechowuje tests[j]=’FaultFree jeżeli oczekuje się że test węzła n_j przeprowadzony przez n_i wyda wartość FaultFree . Wartościami oczekiwanymi mogą być ‘FaultFree’, ‘Faulty’oraz ‘NoTest’ jeżeli oczekuje się że n_i nie testuje n_j.

Pakiety

Struktura danych pakietu przesyłanego pomiędzy węzłami zawiera:

- ***pkt.event*** Tablica znaczników czasu zdarzeń o konstrukcji analogicznej do tablicy *event* w danych węzła. Tablica zawiera pozycje odpowiadające poszczególnym węzłom, jedna pozycja dla każdego węzła. Gdy pakiet dociera do węzła wszystkie pozycje tablicy czasów zdarzeń węzła i pakietu są porównywane ze sobą i ustawiane na wartość większą z dwu porównywanych. Po tym kroku obie tablice zawierają identyczne wartości. W zależności od tego czy i która tablica znaczników była modyfikowana podejmowane są stosowne czynności.
- ***pkt.from*** tablice w której zapisani są poprzednicy każdego węzła w ścieżce propagacji pakietu, zanim pakiet jest wysyłany od korzenia poprzez gałęzie do liści tablica *pkt.from* jest zerowana (wszystkie pola ustawiane są na wartość –1), w czasie oddalania się od węzła korzenia tablica jest sukcesywnie wypełniana, na podstawie tablicy można stwierdzić czy dany węzeł był już odwiedzany przez pakiet;
- ***pkt.topology*** tablice która zawiera potencjalny końcowy przydział testów, w czasie pierwszego przechodzenia drzewa- czyli konstruowania grafu struktury diagnostycznej tablica jest wyzerowana (wszystkie pola ustawiane są na wartość –1). Gdy po pierwszym przebiegu drzewa pakiet z kompletną tablicą *pkt.from* wróci do korzenia, tablica *pkt.from* jest kopiowana do tablicy *pkt.topology*, następnie tablica *pkt.from* jest zerowana (wszystkie pola ustawiane są na wartość –1), i rozpoczyna się drugi przebieg drzewa w celu rozprowadzenia aktualnego przydziału testów zawartego w *pkt.topology* do wszystkich węzłów w grafie.
- ***pkt.istested*** tablice identyfikujące uszkodzony węzeł jest obecnie testowany wg. przydziału. Tablica zapewnia, że tylko jeden i przynajmniej jeden węzeł z pośród wszystkich działających testuje węzły które są uznane za uszkodzone. Węzłem wyznaczanym do testowania węzłów uznanych za uszkodzone jest zawsze węzeł-korzeń ostatniej fazy aktywnej algorytmu.

Struktury danych pakietu

```
Struct pakiet
{
        int sendid                  //identyfikator nadawcy
        int recvid                  //identyfikator odbiorcy
        int rootevent               //znacznik czasu root-a w momencie wysłania
                                    pakietu
        int event[nodenum]          //znaczniki czasów węzłów (do unieważniania
                                    //nieaktualnych pakietów)
        int from[nodenum]           //ścieżka propagacji pakiet.from[i]=j  <=> i po
                                    // raz  pierwszy otrzymał pakiet od j
        int topology[nodenum]       //potencjalny przydział testów (określany dla
                                    //kompletnego pakietu na podstawie tablicy from
–                                   //struktura drzewiasta
        char istested[nodenum]      //istested[i]=TRUE  <=> uszkodzony węzeł i jest
                                    //testowany
} pkt;
```

Faza pasywna. Okresowe testowanie wyznaczonych węzłów

```
PeriodicTest()
{
      dla każdego {i: tests[i]!=Nie_testuj}
      {
      testuj n_i;
      if ((wynik testu n_i) != tests[i])     //gdy wynik testu jest inny niż
                                               oczekiwany
            StartEventPackets();     //rozpocznij fazę aktywną
      }
}
```

W czasie pasywnej fazy algorytmu, przeprowadzane są okresowe testy przez procedurę ***PeriodicTest***, pakiety nie są transmitowane, każdy węzeł posiada tablice *diagnosis* z ostatnimi uaktualnieniami. Uszkodzenie jest wykrywane na podstawie różnic w wynikach otrzymywanych przez poszczególne testy okresowe. Procedura ***StartEventPacket*** jest wywoływana i inicjowana jest faza aktywna.

Faza aktywna

Procedura StartEventPacket zwiększa swoją pozycję w tablicy znaczników czasu węzła oraz inicjuje dane w nowym pakiecie. Pakiet posiada uaktualnioną tablicą *pkt.event* na podstawie tablicy znaczników czasu węzła, wykasowane propozycje przydziałów testów w tablicy *pkt.topology*, ponieważ pakiet nie ma wygenerowanej bieżącej topologii sieci, wykasowana ścieżka propagacji pakietu w tablicy *pkt.from*. Pakiet jest przesyłany dalej do wolnego od uszkodzeń sąsiada n_i, wolny od uszkodzeń sąsiad do którego przesłano pakiet jest oznaczany w tablicy *tests* węzła jako sprawny.

Start fazy aktywnej

```
StartEventPackets()
{
        event[nodeid]++;                //zwiększ znacznik czasu
        pkt.event=event;                //ustaw znacznik czasu w pakiecie na
                                          bieżący
        pkt.topology=-1;                // brak przydziału testów
        pkt.from=-1;                    //ścieżka propagacji – nieznana
        if( istnieje zdatny sąsiad n_i)
        {
                wyślij pakiet do n_i;   //prześlij pakiet
                tests[i]=Zdatny;        //dodaj do testów
        }
}
```

Faza aktywna algorytmu jest kontynuowana w każdym węźle przez procedurę ReceivePacket wykonywaną gdy węzeł odbierze pakiet. W kroku 1 procedury ReceivePacket w przypadku gdy pakiet dotarł do węzła po raz pierwszy rodzic bieżącego węzła jest zapisywany w pakiecie (w tablicy *pkt.from*). W kroku 2 wszystkie pozycje tablicy czasów zdarzeń węzła i pakietu są porównywane ze sobą i ustawiane na wartość większą z dwu porównywanych. Po tym kroku obie tablice zawierają identyczne wartości. W kroku 3 gdy wystąpiły zmiany w tablicy znaczników czasu pakietu, kasowany jest sugerowany przydział testów. W kroku 4 gdy pakiet okaże się najbardziej aktualnym pakietem odebranym przez węzeł, przydział testów jest ustawiany według ścieżki propagacji pakietu. Stosuje się strategię „dogrywki" (tie-breaking), w przypadku dwu pakietów z jednakowymi tablicami znaczników zdarzeń pochodzących w różnych węzłów-korzeni. W kroku 5 pakiet jest propagowany zgodnie z algorytmem przeszukiwania „najpierw w głąb" (deap-first search). Krok 6 jest wykonywany gdy pakiet gdy pakiet jest kompletny, pakiet jest wtedy propagowany do wszystkich wolnych od uszkodzeń węzłów oraz zwracany do węzła korzenia. Na zakończenie w przypadku gdy pakiet był modyfikowany (pkt.topology= -1), nowy pakiet z aktualną tablicą zdarzeń jest propagowany w celu podjęcia próby ustawienia przydziału testów.

Złożoność obliczeniowa algorytmu

Algorytm wykonuje się w dwóch fazach: **pasywnej** i **aktywnej**. W fazie pasywnej wykonuje się testy według ustalonych przydziałów. Przydział testów gwarantuje, że każdy zdarzenie związane z awarią węzła lub połączenia zostanie wykryte przez co najmniej jeden wolny od uszkodzeń węzeł. Faza aktywna jest inicjowana w przypadku wykrycia zdarzenia związanego z awarią. Sprawne węzły przeprowadzają proces konstruowania nowego przydziału testów oraz aktualizują dane o stanie systemu (diagnozę systemu). Gdy przydział testów jest skonstruowany algorytm powraca do fazy pasywnej. Algorytm gwarantuje wykrywanie uszkodzeń zarówno w fazie pasywnej, jak i aktywnej. Faza aktywna wymaga maksymalnie do $\boldsymbol{O(N\log_2 N)}$ komunikatów oraz posiada ***opóźnienie diagnozy O(N).***