Netgraph w systemie FreeBSD

Wojciech A. Koszek
dunstan@FreeBSD.czest.pl
MeetBSD 2005
Kraków 17-19.06.2005

Czym jest Netgraph?

- Podsystemem jądra FreeBSD
- Rozszerzeniem możliwości i funkcjonalności kodu obsługi sieci

Dlaczego “Netgraph”?

- Ponieważ we FreeBSD już od wersji 3.4-RELEASE stos sieciowy można przedstawić w postaci grafu!

Netgraph jest:

- Modularny
- Rozszerzalny
- Nie narzuca ograniczeń związanych z konfiguracją.

Możliwości:

- Dostęp do najniższych warstw podsystemu obsługi sieci
- Implementacja całych rodzin protokołów w podsystemie Netgraph

Bluetooth we FreeBSD

- Przykład protokołu zaimplementowanego w Netgraph
- Przykład sterownika zaimplementowanego w Netgraph

Netgraph: używane terminy

- węzeł (*ang. node*)
- uchwyt (*ang. hook*)
- krawędź (ang. edge)

Node: instancja węzła w Netgraph

- Pełni ściśle określoną funkcję:
  - odbiera/wysyła dane odpowiednio je przekształcając
  - odbiera i wysyła dodatkowe dane
  - może posiadać swoją nazwę (użyteczne w konfiguracji)

Hook: uchwyt w Netgraph

- Służy do łączenia struktury węzłów
- Jest pośrednikiem między węzłami, po którym poruszają się dane

Nazwy uchwytów:

- Nazwy funkcjonalne („left2right" w ng_tee(4), „lower" i „downstream" w *ng_ether(4)*)
- Możliwość wyboru nazw dla uchwytów (*ng_echo(4)*)

Egde: "krawędź grafu przepływu":

- Połączenie poprzez którym dane poruszają się w obu kierunkach
- Krawędź powstaje poprzez połączenie dwóch instancji węzła, w którym uchwyt odgrywa rolę pośrednika

Netgraph: przepływ danych

- Dane mogą poruszać się w obu kierunkach po krawędziach grafu
- Wykorzystuje istniejący kod sieciowy znany z *BSD (bufory *mbuf_*/clusters*)

Funkcjonalność jako moduły jądra

- Podsystem ładowany jako KLD:
- Każdy węzeł ładowany jako KLD
- Narzędzia wchodzą w skład *base-system'u*

Krótki przegląd funkcjonalności:

- ng_UI.ko
- ng_async.ko
- ng_atmllc.ko
- ng_base.ko
- ng_bpf.ko
- ng_bridge.ko
- ng_cisco.ko
- ng_device.ko
- ng_echo.ko
- ng_eiface.ko

..krótki przegląd funkcjonalności..

- ng_etf.ko
- ng_ether.ko
- ng_fec.ko
- ng_frame_relay.ko
- ng_gif.ko
- ng_gif_demux.ko
- ng_hole.ko
- ng_hub.ko
- ng_iface.ko
- ng_ip_input.ko

..krótki przegląd funkcjonalności..

- ng_ipfw.ko
- ng_ksocket.ko
- ng_l2tp.ko
- ng_lmi.ko
- ng_mppc.ko
- ng_one2many.ko
- ng_parse.ko
- ng_ppp.ko
- ng_pppoe.ko
- ng_pptpgre.ko

..krótki przegląd funkcjonalności..

- ng_rfc1490.ko
- ng_sample.ko
- ng_socket.ko
- ng_source.ko
- ng_split.ko
- ng_sppp.ko
- ng_tee.ko
- ng_tty.ko
- ng_vjc.ko
- ng_vlan.ko

Komunikacja z węzłami:

- Komunikaty wysyłane z przestrzeni użytkownika
- Komunikaty nie przechodzą przez graf przepływu - są dostarczane osobno, bezpośrednio do instancji węzła
- Wykorzystywane przez narzędzia z przestrzeni użytkownika

Komunikacja z węzłami (2)

- Instancje węzła wykorzystują:
  - standardowe komunikaty Netgraph
- Komunikaty dodatkowe
  - zależne od rodzaju węzła

ngctl(8): adresacja węzłów

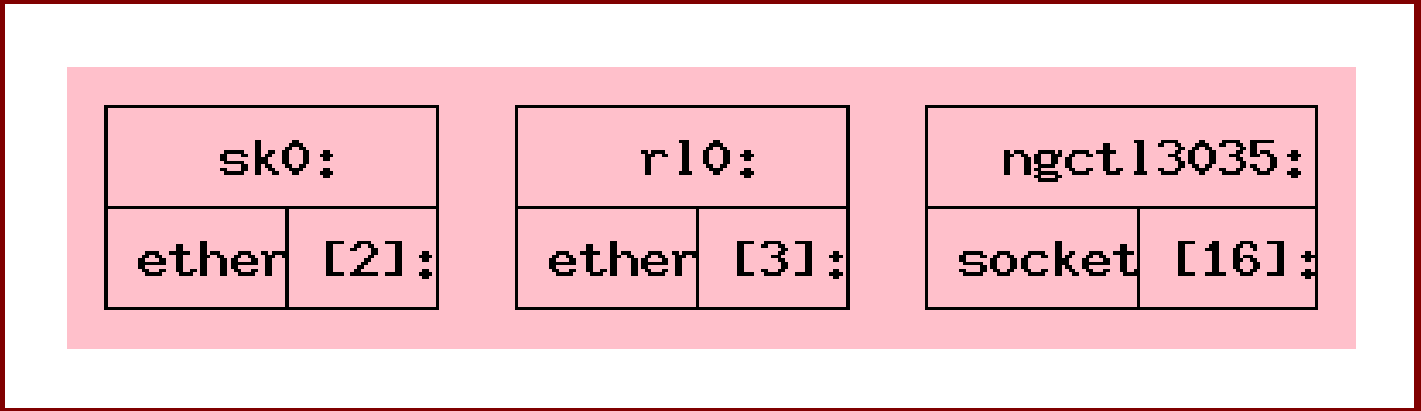
- Węzły posiadają swój model adresowania:
  - “.” -- węzeł obecny
  - ngeth0: -- dostęp do węzła ng0:
  - ngeth0:ether -- dostęp do uchwytu *"ether"* węzła *"ng0"*

Narzędzia:

- *ngctl(8)*
  - Główne narzędzie konfiguracyjne
- *nghook(8)*
  - Umożliwia podłączenie się do określonego uchwytu węzła.
- *libnetgraph(3)*
  - Komunikacja z przestrzeni użytkownika

ngctl(1)

```
+ list
There are 3 total nodes:
  Name: ngctl2979   Type: socket   ID: 00000013   Num hooks: 0
  Name: rl0         Type: ether    ID: 00000003   Num hooks: 0
  Name: sk0         Type: ether    ID: 00000002   Num hooks: 0
+
```



Konfiguracja również przy pomocy Netgraph?

- ngctl(4) komunikuje się z jądrem poprzez socket (*AF_NETGRAPH*)
- socket od strony jądra udostępniany przez *ng_socket(4)*

Przykłady – *ng_echo(4)* użyteczne w testowaniu

- Odbiera dane i komunikaty, którymi następnie odpowiada
- Dobry jako przykład - wszystkie operacje na innych węzłach wykonuje się analogicznie

Tworzenie nowej instancji węzła:

mkpeer <adres> <typ> <uchwyt_adres> <uchwyt_typ>

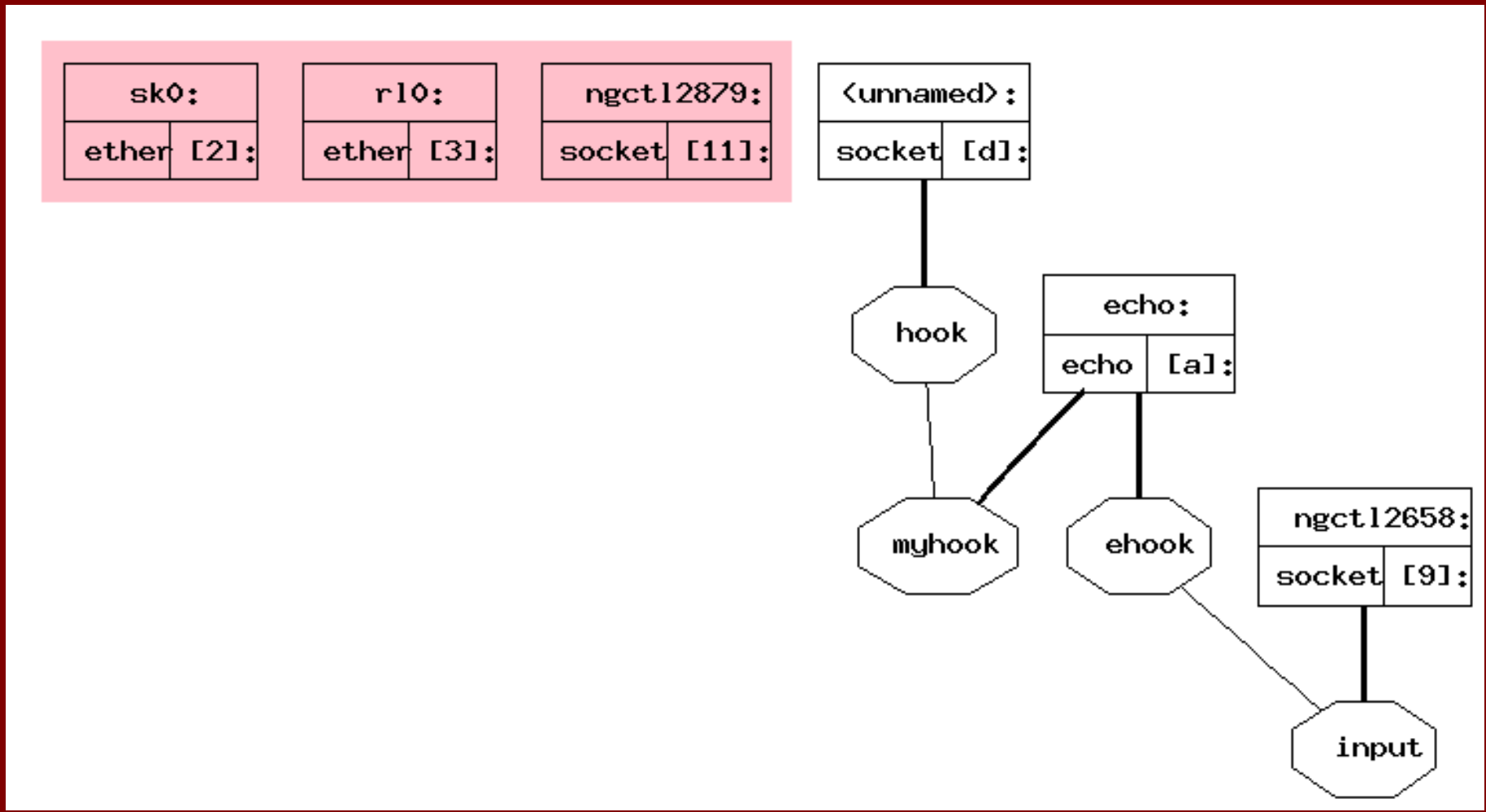
```
+ mkpeer . echo input ehook
+ list
There are 4 total nodes:
  Name: <unnamed>  Type: echo   ID: 0000000a  Num hooks: 1
  Name: ngctl2658  Type: socket ID: 00000009  Num hooks: 1
  Name: rl0        Type: ether  ID: 00000003  Num hooks: 0
  Name: sk0        Type: ether  ID: 00000002  Num hooks: 0
+ name .:input echo
+ list
There are 4 total nodes:
  Name: echo       Type: echo   ID: 0000000a  Num hooks: 1
[..]
+
```

Tworzenie nowej instancji węzła (2)

- Instancja istnieje gdy:
  - Reprezentuje pewien permanentny obiekt w jądrze
    - Interfejs
    - Urządzenie
- Istnieje przynajmniej jedna aktywna krawędź

Co udało się uzyskać?

```
# nghook -a echo: myhook
MeetBSD 2005
0000:  4d 65 65 74 42 53 44 20 32 30 30 35 0a   MeetBSD 2005.
```

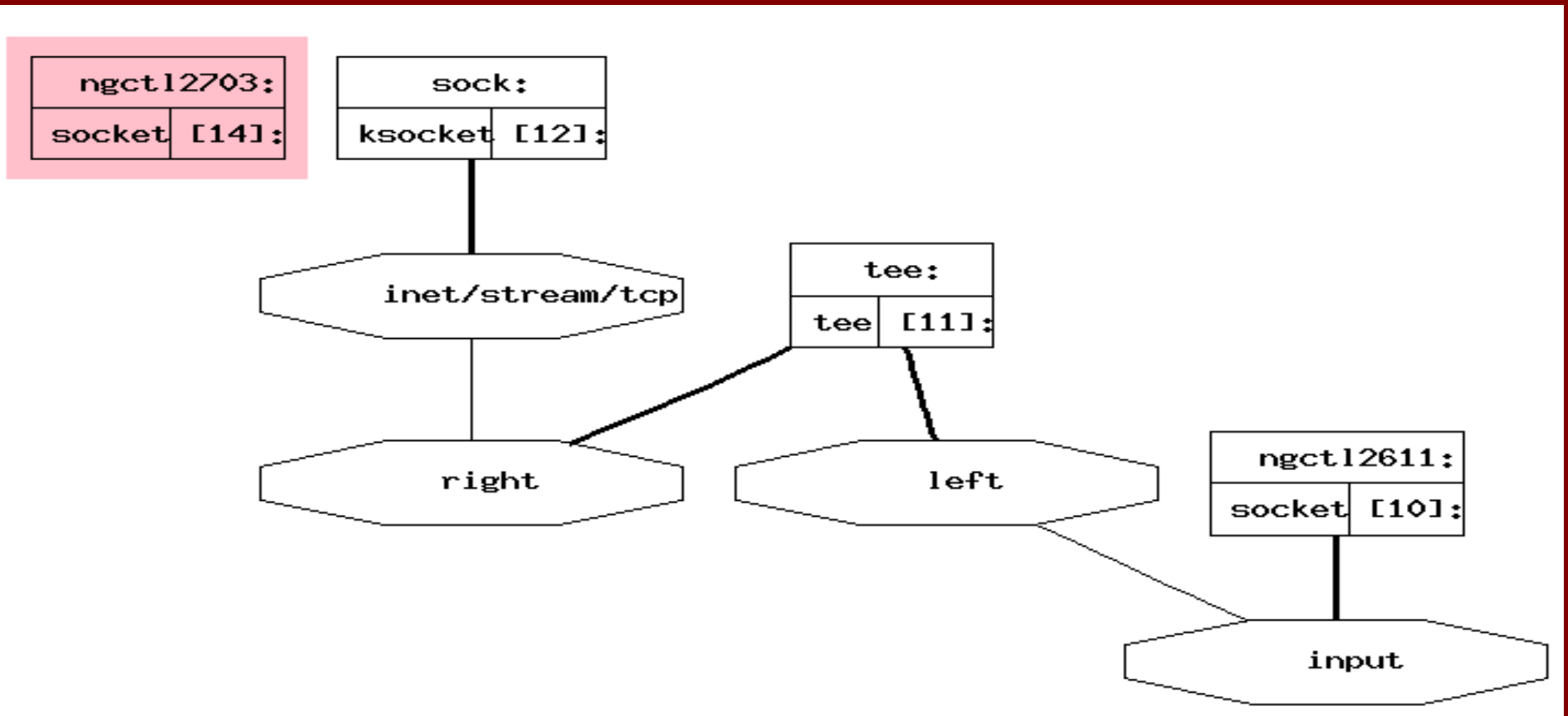


ng_tee (4): użyteczne w śledzeniu ruchu

- Uchwyty:
  - “left” oraz “right”, między którymi przechodzi ruch sieciowy
  - “left2right” & “right2left”, na których pojawiają się dane przechodzące w odpowiednim kierunku

ng_tee (4): konfiguracja

+ mkpeer . tee input left
+ name .:input tee
+ mkpeer tee: ksocket right inet/stream/tcp
+ name tee:right sock



Graficzna reprezentacja grafu, Graphviz w */usr/{ports|pkgsrc}*:

```
+ ngctl dot
graph netgraph {
  [..]
        edge [ weight = 1.0 ];
        node [ shape = record, fontsize = 12 ]
        {
    [..]
        }
    [..]
};
```

Obecny stan prac nad Netgraph'em:

- Pełne wsparcie dla SMP:
  - Netgraph wykorzystuje mechanizmy synchronizacji.
  - Jednak nadal pod kontrolą Giant'a
  - Potrzebne wykorzystanie dobrego mechanizmu sygnalizacji zdarzeń - migracja do *EVENTHANDLER(9)*?

Koniec

- Bardzo dziękuję za uwagę. Zapraszam do zadawania pytań