Zadání úlohy Rich Club - 3rc

Cíl úlohy:

- Získat schopnost zpracovat data reálné sítě pomocí netflow záchytu.
- Naučit se načítat netflow data.
- Naučit se dekomponovat netflow data na podsítě služeb.
- Naučit se detekovat případný klub bohatých.

Zadání:

1. Připravte si vstupní data:
   a) poskytnutý netflow záchyt dekompresujte do Vámi vybraného adresáře
   b) zjistěte, podle kterého protokolu netflow jsou data zachycena (tj. číslo netflow protokolu) a která endianita (pořadí bytů) je použita
   c) převeďte si vstupní data do formátu, které umí zpracovávat Vámi vybrané knihovny
2. Data načtěte do paměti a zjistěte základní charakteristiky komunikační sítě:
   a) zjistěte
      - celkový počet netflow záznamů,
      - celkový počet přenesených paketů,
      - celkový počet zdrojových IP adres,
      - celkový počet cílových IP adres,
      - distribuci použití protokolů transportní vrstvy
   b) vyberte pouze netflow týkající se protokolů TCP a UDP
3. Vytvořte vizualizaci komunikační sítě mezi IP adresami, tj. dvě IP adresy komunikují, pokud byl mezi přenesen alespoň jeden paket.
4. Spočtěte distribuci cílových portů transportního provozu v celém rozsahu 0 - 65535 a v rozsahu systémových portů (0-1023).
5. Pro každý systémový cílový port s počtem netflow výskytů vyšším než 200 a počtem příslušných cílových adres vyšším než 10:
   a) vytvořte podgraf komunikační sítě mezi zdrojovými IP adresami a cílovými IP adresami kontaktovanými právě na daném cílovém portu včetně jeho vhodné vizualizace.

b) Spočtěte distribuci stupňů v takové síti.

c) Určete koeficient bohatého klubu pro každou hodnotu stupně v takové síti a zobrazte tuto závislost.

d) Pro hodnotu stupně, která generuje nejvyšší koeficient bohatého klubu s nejvyšším počtem IP adres, vygenerujte příslušný podgraf včetně jeho vizualizace, pokud jeho řád je alespoň 5. Diskutujte role IP adres patřících do takového klubu.

Doporučení:

1. Doporučujeme použít jazyk Python a knihovny NetworkX a Matplotlib
2. K vytvoření a nastavení cesty pro generované diagramy doporučujeme použít funkce

```
def SetOutPN(subFolder):
  #====================
  # Open a necessary output infrastructure
  outPN = subFolder
  if not os.path.exists(outPN):
    os.makedirs(outPN)
  return outPN
```

3. K zobrazeni histogramu doporučujeme vyjít z funkce

```
def Histogram(sPlotIndex, x = None, xLabel = '', yLabel = '', num_bins = 100, log = False,
              normed = 0, colorLabel = None, alpha = None, **kwargs):

#=============================================================================================
  fig = plt.figure()
  fig.set_size_inches(4, 3) # width and height in inches
  ax = plt.subplot(sPlotIndex)
  n, bins, patches = plt.hist(x, num_bins, normed=normed, log = log, facecolor='green',
alpha=0.5)
  if log:
    plt.xscale('log')
    plt.ylim(ymin=0.5)
  if xLabel: plt.xlabel(xLabel)
  if yLabel: plt.ylabel(yLabel)
```

4. V případě, že Vám diagramy z různých vizualizací pomocí knihovny matplotlib neočekávaným způsoben interagují, je možné po uložení diagramu doporučujeme resetovat knihovnu matplotlib pomocí funkcí použitých v následující funkci (Upozornění: výběr funkcí je však potřeba přizpůsobit Vámi používané verzi knihovny).

```
def PltClear():
  #============
  plt.clf()
  fig = plt.gcf()
  plt.close(fig)
  plt.close('all')
```

5. Před zapsáním diagram do souboru doporučujeme použít funkci `plt.tight_layout()`

6. Pro zápis diagram do souboru doporučujeme použít funkci `plt.savefig`

7. Pro získání seznamu všech hodnot stupňů uzlu v daném grafu `myGraph` doporučuje využít konstrukce
   `list(nx.degree(myGraph).values())`

8. Pro zjištění informací o surových netflow datech a jejich konverzi na vhodný formát, např. CSV, můžete použít např. Sadu nástrojů flow-tools (na Linuxu instalací „apt-get install flow-tools“, zvláště pak nástroje flow-header a flow-export. Zvažte, zda není výhodné použít při exportu techniku pipe, kterou transformovaná data opět kompresujete, např. Programem „gzip“.
9. K vyhledání všech souborů v daném adresáři můžete v jazyku Python použít funkci „glob“ z modulu „glob“.
10. K načítání zazipovaných dat můžete v Pythonu použít konstrukci série načítacích funkcí
```
with open(fN, "rb") as inBinF:
  inGZF  = TextCodecWrapper(gzip.GzipFile(fileobj = inBinF, mode = 'r'),
                            "utf-8", errors='ignore')
  dataReader = csv.reader(inGZF, delimiter=',', quotechar='"')
```
    kde „TextCodecWrapper“ je namapován v Python 2.7 na
```
  from codecs import EncodedFile as TextCodecWrapper
```
    a v Python 3.* na
```
  from io import TextIOWrapper as TextCodecWrapper
```
11. K zobrazení funkčních závislostí můžete použít funkce „bar“ či „plot“ z modulu „matplotlib“.
12. Pro lepší zobrazování vizualizací sítě zvažte, zda není síť lepší exportovat do souboru GML pomocí funkce „write_gml“ z modulu „networkx“. Soubor GML pak lze zobrazit v komunitní verze editoru yEd (http://www.yWorks.com) či nástroje „Gephi“ (https://gephi.org/).
13. K získání podgrafu indukovaného podmnožinou uzlů můžete použít funkci „subgraph“ z modulu „networkx“.
14. K získání unikátních hodnot lze v Python použít množiny či slovníky.
15. K spočítání koeficientu klubu bohatých pro všechny stupně lze použít funkci „rich_club_coefficient” z modulu `“networkx“.`
16.