Schedule change

- Class 9 Measuring traffic of the Internet (11/18, Friday) 9:25-10:55 e11
- Class 10 Hot topics (11/18, Friday) 11:10-12:40 e11
- Class 11 Measuring time series of the Internet (11/30)
- NO Class (12/7)
review of previous class

Class 5 Measuring the structure of the Internet

- Internet architecture
- network layers
- topologies
- graph theory
- exercise: topology analysis
today’s topics

Class 6 Measuring the characteristics of the Internet

- delay, packet loss, jitter
- correlation and multivariate analysis
- principal component analysis
- exercise: correlation analysis
measurement metrics of the Internet

measurement metrics

- link capacity, throughput
- delay
- jitter
- packet loss rate

methodologies

- active measurement: injects measurement packets (e.g., ping)
- passive measurement: monitors network without interfering in traffic
  - monitor at 2 locations and compare
  - infer from observations (e.g., behavior of TCP)
  - collect measurements inside a transport mechanism
delay measurement

- delay components
  - delay = propagation delay + queueing delay + other overhead
  - if not congested, delay is close to propagation delay

- methods
  - round-trip delay
  - one-way delay requires clock synchronization

  - average delay
  - max delay: e.g., voice communication requires $< 400\, ms$
  - jitter: variations in delay
some delay numbers

- packet transmission time (so called wire-speed)
  - 1500 bytes at 10Mbps: 1.2msec
  - 1500 bytes at 100Mbps: 120usec
  - 1500 bytes at 1Gbps: 12usec

- speed of light in fiber: about 200,000 km/s
  - 100km round-trip: 1 msec
  - 20,000km round-trip: 200msec

- satellite round-trip delay
  - LEO (Low-Earth Orbit): 200 msec
  - GEO (Geostationary Orbit): 600msec
packet loss measurement

packet loss rate

- loss rate is enough if packet loss is random...
- in reality,
  - bursty loss: e.g., buffer overflow
  - packet size dependency: e.g., bit error rate in wireless transmission
the Internet End-to-end Performance Measurement (IEPM) project by SLAC

- using ping to measure rtt and packet loss around the world
  - http://www-iepm.slac.stanford.edu/pinger/
  - started in 1995
  - over 600 sites in over 125 countries
pingER project monitoring sites

- monitoring (red), beacon (blue), remote (green) sites
  - beacon sites are monitored by all monitors

from pingER web site
pingER project monitoring sites in east asia

- monitoring (red) and remote (green) sites

from pingER web site
pingER packet loss

- packet loss observed from N. Ameria
- exponential improvement in 10 years

Packet Loss Seen From N. America

From SLAC PingER Project
http://www-iepm.slac.stanford.edu/pinger

4.6% improvement/year

from pingER web site
pinger minimum rtt

- minimum rtts observed from N. America
- gradual shift from satellite to fiber in S. Asia and Africa

From SLAC PingER Project
http://www-ping.slc.stanford.edu/pinger

Min RTT from N. America to World Regions

from pingER web site
variables in data set

- univariate analysis
  - explores a single variable in a data set, separately
- multivariate analysis
  - looks at more than one variables at a time
  - enabled by computers
  - finding hidden trends (data mining)
scatter plots

- explores relationships between 2 variables
  - X-axis: variable X
  - Y-axis: corresponding value of variable Y
- you can identify
  - whether variables X and Y related
    - no relation, positive correlation, negative correlation
  - whether the variation in Y changes depending on X
  - outliers
- examples: positive correlation 0.7 (left), no correlation 0.0 (middle), negative correlation -0.5 (right)
correlation

- covariance:
  \[ \sigma_{xy}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \]

- correlation coefficient:
  \[ \rho_{xy} = \frac{\sigma_{xy}^2}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}} \]

- correlation coefficient: the covariance of 2 variables normalized by their product of their standard deviations, a value between −1 and +1 inclusive.
- sensitive to outliers. so, you should use a scatter plot to observe outliers.
- correlation and causality
  - correlation does not imply causal relationship
    - third factor C causes both A and B
    - coincidence
correlation and multivariate analysis

multivariate analysis: statistical methods to analyze more than one variable at a time

- visualization of relationship
  - cluster analysis: calculate distance (or similarity) between variables, and assign the variables into groups (or clusters)

- demensionality reduction
  - principal component analysis: a technique to reduce the number of variables
principal component analysis; PCA

purpose of PCA

- convert a set of possibly correlated variables into a smaller set of uncorrelated variables

PCA can be solved by eigenvalue decomposition of a covariance matrix

applications of PCA

- dimensionality reduction
  - sort principal components by contribution ratio, components with small contribution ratio can be ignored

- principal component labeling
  - find means of produced principal components

notes:

- PCA just extracts components with large variance
  - not simple if axes are not in the same unit

- a convenient method to automatically analyze complex relationship, but it does not explain the complex relationship
PCA: intuitive explanation

a view of coordinate transformation using a 2D graph

- draw the first axis (the 1st PCA axis) that goes through the centroid, along the direction of the maximal variability
- draw the 2nd axis that goes through the centroid, is orthogonal to the 1st axis, along the direction of the 2nd maximal variability
- draw the subsequent axes in the same manner

For example, “height” and “weight” can be mapped to “body size” and “slimness”. we can add “sitting height” and “chest measurement” in a similar manner
principal components can be found as the eigenvectors of a covariance matrix.

Let $X$ be a $d$-dimensional random variable. We want to find a $d \times d$ orthogonal transformation matrix $P$ that converts $X$ to its principal components $Y$.

$$Y = P^\top X$$

Solve this equation, assuming $\text{cov}(Y)$ being a diagonal matrix (components are independent), and $P$ being an orthogonal matrix. ($P^{-1} = P^\top$)

The covariance matrix of $Y$ is

$$\text{cov}(Y) = \text{E}[YY^\top] = \text{E}[(P^\top X)(P^\top X)^\top] = \text{E}[(P^\top X)(X^\top P)]$$

$$= P^\top \text{E}[XX^\top]P = P^\top \text{cov}(X)P$$

Thus,

$$P \text{cov}(Y) = PP^\top \text{cov}(X)P = \text{cov}(X)P$$

Rewrite $P$ as a $d \times 1$ matrix:

$$P = [P_1, P_2, \ldots, P_d]$$

Also, $\text{cov}(Y)$ is a diagonal matrix (components are independent)

$$\text{cov}(Y) = \begin{bmatrix}
\lambda_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \lambda_d
\end{bmatrix}$$

This can be rewritten as

$$[\lambda_1 P_1, \lambda_2 P_2, \ldots, \lambda_d P_d] = [\text{cov}(X)P_1, \text{cov}(X)P_2, \ldots, \text{cov}(X)P_d]$$

For $\lambda_i p_i = \text{cov}(X)p_i$, $p_i$ is an eigenvector of the covariance matrix $X$.

Thus, we can find a transformation matrix $P$ by finding the eigenvectors.
previous exercise: Dijkstra algorithm

- read a topology file, and compute shortest paths

% cat topology.txt
a - b 5
a - c 8
b - c 2
b - d 1
b - e 6
c - e 3
d - e 3
c - f 3
e - f 2
d - g 4
e - g 5
f - g 4
% ruby dijkstra.rb -s a topology.txt
a: (0) a
b: (5) a b
c: (7) a b c
d: (6) a b d
e: (9) a b d e
f: (10) a b c f
g: (10) a b d g
%
previous exercise: sample code (1/4)

# dijkstra’s algorithm based on the pseudo code in the wikipedia
# http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
#
require 'optparse'

source = nil # source of spanning-tree

OptionParser.new { |opt|
  opt.on('-s VAL') { |v| source = v}
  opt.parse!(ARGV)
}

INFINITY = 0x7fffffff # constant to represent a large number
previous exercise: sample code (2/4)

```ruby
# read topology file and initialize nodes and edges
# each line of topology file should be "node1 (-|->) node2 weight_val"
nodes = Array.new  # all nodes in graph
edges = Hash.new  # all edges in graph
ARGV.each_line do |line|
  s, op, t, w = line.split
  next if line[0] == '#' || w == nil
  unless op == '-' || op == '->'
    raise ArgumentError, "edge_type should be either '-' or '->'"
  end
  weight = w.to_i
  nodes << s unless nodes.include?(s)  # add s to nodes
  nodes << t unless nodes.include?(t)  # add t to nodes
  # add this to edges
  if (edges.has_key?(s))
    edges[s][t] = weight
  else
    edges[s] = {t=>weight}
  end
  if (op == "-")  # if this edge is undirected, add the reverse directed edge
    if (edges.has_key?(t))
      edges[t][s] = weight
    else
      edges[t] = {s=>weight}
    end
  end
end
# sanity check
if source == nil
  raise ArgumentError, "specify source_node by '-s source'"
end
unless nodes.include?(source)
  raise ArgumentError, "source_node(#{source}) is not in the graph"
end
```
# create and initialize 2 hashes: distance and previous
dist = Hash.new # distance for destination
text = Hash.new # previous node in the best path

nodes.each do |i|
    dist[i] = INFINITY # Unknown distance function from source to v
text[i] = -1 # Previous node in best path from source
end

# run the dijkstra algorithm

dist[source] = 0 # Distance from source to source

while (nodes.length > 0)
    # u := vertex in Q with smallest dist[]
    u = nil

    nodes.each do |v|
        if (!u) || (dist[v] < dist[u])
            u = v
        end
    end

    if (dist[u] == INFINITY)
        break # all remaining vertices are inaccessible from source
    end

    nodes = nodes - [u] # remove u from Q

    # update dist[] of u’s neighbors
    edges[u].keys.each do |v|
        alt = dist[u] + edges[u][v]
        if (alt < dist[v])
            dist[v] = alt
text[v] = u
        end
    end
end
# print the shortest-path spanning-tree
dist.sort.each do |v, d|
  print "#{v}: " # destination node
  if d != INFINITY
    print "(#{d}) " # distance
    # construct path from dest to source
    i = v
    path = "#{i}"
    while prev[i] != -1 do
      path.insert(0, "#{prev[i]} ") # prepend previous node
      i = prev[i]
    end
    puts "#{path}" # print path from source to dest
  else
    puts "unreachable"
  end
end
exercise: correlation

request-table in Class 4:

- use the 5-minute bin output from the previous class
- focus on the request counts
- use 12 5-minute bins for an hour as 12 samples per hour
exercise: correlation (cont’d)

an hourly request table

- row: hourly data (0 .. 23)
- column: hour samples(00 05 10 ... 55) mean stddev

Correlation computation

- use 12 5-minute bins as 12 time-series
- compute correlation coefficient among time slots (columns)

<table>
<thead>
<tr>
<th>#hour</th>
<th>00</th>
<th>05</th>
<th>10</th>
<th>...</th>
<th>55</th>
<th>mean</th>
<th>stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4123</td>
<td>3963</td>
<td>3871</td>
<td>...</td>
<td>3987</td>
<td>4046.8</td>
<td>102.3</td>
</tr>
<tr>
<td>1</td>
<td>4068</td>
<td>3871</td>
<td>3838</td>
<td>...</td>
<td>3760</td>
<td>3774.9</td>
<td>106.2</td>
</tr>
<tr>
<td>2</td>
<td>3833</td>
<td>3755</td>
<td>3580</td>
<td>...</td>
<td>3628</td>
<td>3703.6</td>
<td>219.0</td>
</tr>
<tr>
<td>3</td>
<td>3614</td>
<td>3433</td>
<td>3418</td>
<td>...</td>
<td>3462</td>
<td>3515.5</td>
<td>86.2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>4724</td>
<td>4790</td>
<td>4757</td>
<td>...</td>
<td>4893</td>
<td>4882.2</td>
<td>113.4</td>
</tr>
<tr>
<td>23</td>
<td>4922</td>
<td>4932</td>
<td>4889</td>
<td>...</td>
<td>4188</td>
<td>4818.9</td>
<td>203.8</td>
</tr>
</tbody>
</table>
### correlation matrix

Compute correlation matrix for the 12 time-series data

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>05</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1.00</td>
<td>0.98</td>
<td>0.98</td>
<td>0.97</td>
<td>0.95</td>
<td>0.92</td>
<td>0.87</td>
<td>0.85</td>
<td>0.89</td>
<td>0.86</td>
<td>0.89</td>
<td>0.87</td>
</tr>
<tr>
<td>05</td>
<td>0.98</td>
<td>1.00</td>
<td>0.99</td>
<td>0.98</td>
<td>0.97</td>
<td>0.95</td>
<td>0.92</td>
<td>0.88</td>
<td>0.90</td>
<td>0.90</td>
<td>0.92</td>
<td>0.90</td>
</tr>
<tr>
<td>10</td>
<td>0.98</td>
<td>0.99</td>
<td>1.00</td>
<td>0.99</td>
<td>0.98</td>
<td>0.96</td>
<td>0.93</td>
<td>0.91</td>
<td>0.92</td>
<td>0.91</td>
<td>0.93</td>
<td>0.92</td>
</tr>
<tr>
<td>15</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>1.00</td>
<td>0.98</td>
<td>0.97</td>
<td>0.95</td>
<td>0.93</td>
<td>0.95</td>
<td>0.94</td>
<td>0.96</td>
<td>0.94</td>
</tr>
<tr>
<td>20</td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
<td>0.98</td>
<td>1.00</td>
<td>0.99</td>
<td>0.97</td>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>25</td>
<td>0.92</td>
<td>0.91</td>
<td>0.91</td>
<td>0.95</td>
<td>0.97</td>
<td>1.00</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>30</td>
<td>0.87</td>
<td>0.87</td>
<td>0.86</td>
<td>0.91</td>
<td>0.95</td>
<td>1.00</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>35</td>
<td>0.85</td>
<td>0.85</td>
<td>0.84</td>
<td>0.89</td>
<td>0.91</td>
<td>1.00</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>40</td>
<td>0.85</td>
<td>0.85</td>
<td>0.84</td>
<td>0.89</td>
<td>0.91</td>
<td>1.00</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>45</td>
<td>0.84</td>
<td>0.84</td>
<td>0.83</td>
<td>0.87</td>
<td>0.96</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.99</td>
<td>1.00</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>50</td>
<td>0.86</td>
<td>0.87</td>
<td>0.84</td>
<td>0.90</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>55</td>
<td>0.87</td>
<td>0.89</td>
<td>0.88</td>
<td>0.91</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.94</td>
<td>1.00</td>
</tr>
</tbody>
</table>
#!/usr/bin/env ruby

# read request-table.txt
re = /\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+/g
hourly = Array.new(24){ Array.new(12) }
ARGF.each_line do |line|
  if re.match(line)
    for min in 0 .. 11
      hourly[$1.to_i][min] = Regexp.last_match(min + 2).to_i
    end
  end
end

means = Array.new(12)
for min in 0 .. 11
  mean = 0
  for hour in 0 .. 23
    mean += hourly[hour][min]
  end
  means[min] = Float(mean) / 24
end
cc_matrix = Array.new(12){ Array.new(12) }
for m0 in 0 .. 11
  for min in 0 .. 11
    cov = 0
    sum_dx2 = sum_dy2 = 0
    for hour in 0 .. 23
      x = hourly[hour][m0]
      y = hourly[hour][min]
      cov += (x - means[m0]) * (y - means[min])
      sum_dx2 += (x - means[m0])**2
      sum_dy2 += (y - means[min])**2
    end
    cc_matrix[m0][min] = Float(cov) / Math.sqrt(sum_dx2 * sum_dy2)
  end
end

for m0 in 0 .. 11
  for min in 0 .. 11
    printf "%.3f ", cc_matrix[m0][min]
  end
  print "\n"
end
Class 6 Measuring the characteristics of the Internet

- delay, packet loss, jitter
- correlation and multivariate analysis
- principal component analysis
- exercise: correlation analysis
Class 7 Measuring the diversity and complexity of the Internet (11/9)

- sampling
- statistical analysis
- histogram
- exercise: histogram, CDF