Analyzing Simulation Result
～awk, gnuplot～

Practice 2

Information and Communications Technology

Internet Engineering

Using awk for data processing

- Possible to extract data
- Process every line of read data
- Similar to C

- Structure

```
% awk 'Pattern {Action}' Input data file
```

If condition satisfy do the action

If condition command

Simple awk command example

weather.dat

2 column ($2) : Location
3 column ($3) : Temperature
4 column ($4) : Humid

札幌	Sapporo	9.1	84
青森	Aomori	12.1	76
秋田	Akita	13.6	71
盛岡	Morioka	12.7	69
仙台	Sendai	15.6	64
山形	Yamagata	14.2	69
福島	Fukushima	16.7	57
水戸	Mito	16.4	70
宇都宮	Utsunomiya	16.0	76
前橋	Maebashi	17.3	65
熊谷	Kumagaya	17.8	71
東京	Tokyo	18.8	61
⋮			
鹿児島	Kagoshima	19.7	65
那覇	Naha	24.3	79

Data from weather forecast homepage

- Select data when Location is Sendai

```
% awk ‘$2 == “Sendai”’ weather.dat
 – ‘$2 = “Sendai”’?
```

- Select data where humid is 61

```
% awk ‘$4 == 61’ weather.dat
```

- Extract data where temperature below 16

```
% awk ‘$3 <= 16’ weather.dat
```

- Extract data where temperature between 20 and 80

```
% awk ‘$3>20 && $4<80’ weather.dat
```

- Enable operator

```
+, -, *, /, %
<, <=, ==, !=, >, >=
&&, ||
```

Count number of lines

- wc (Word Count)
  - Display number of lines・ word・bytes (size) of a file

```
% wc weather.dat
 47  188 1833 weather.dat
```

- Only display number of lines

```
% wc -l weather.dat
```

Redirection and pipe

- Redirection: save the output of awk to a file
  - Data which have temperature larger than is saved in a file 16

```
% awk '$3 > 16' weather.dat > file
```

  - Number of lines is ?

```
% wc -l file
```

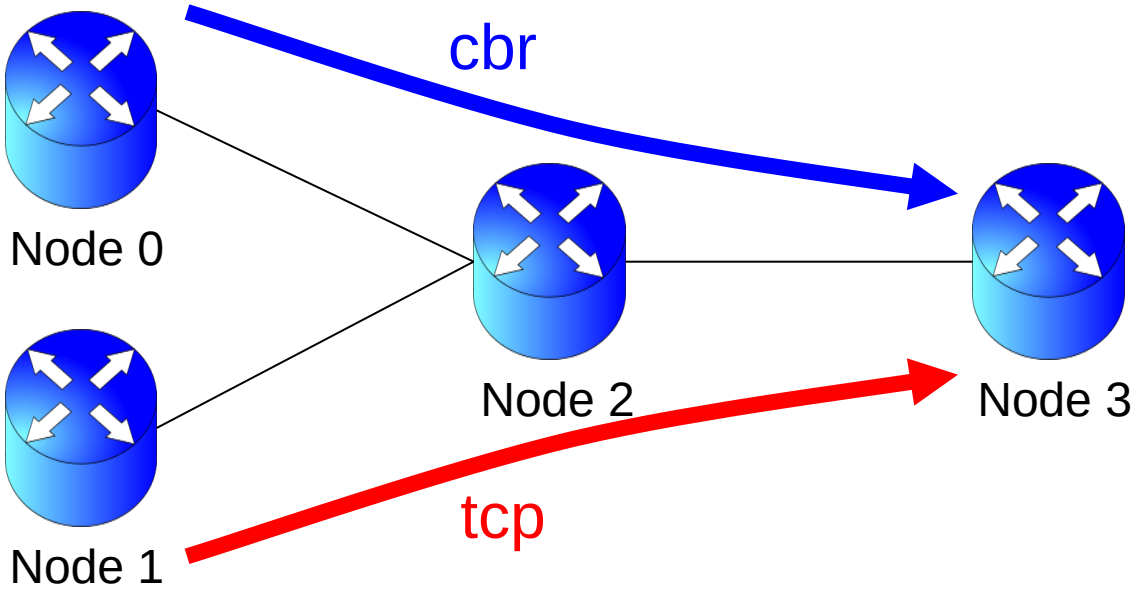
- Pipe: out put of awk command is used as input of other command
  - Number of lines which have temperature bigger than 16

```
% awk '$3 > 16' weather.dat | wc -l
```

Output data of network simulation

- Execute sample.tcl

```
% ns sample.tcl
```



- out.tr, out.tcp, out.nam is created

out.tr (extract)

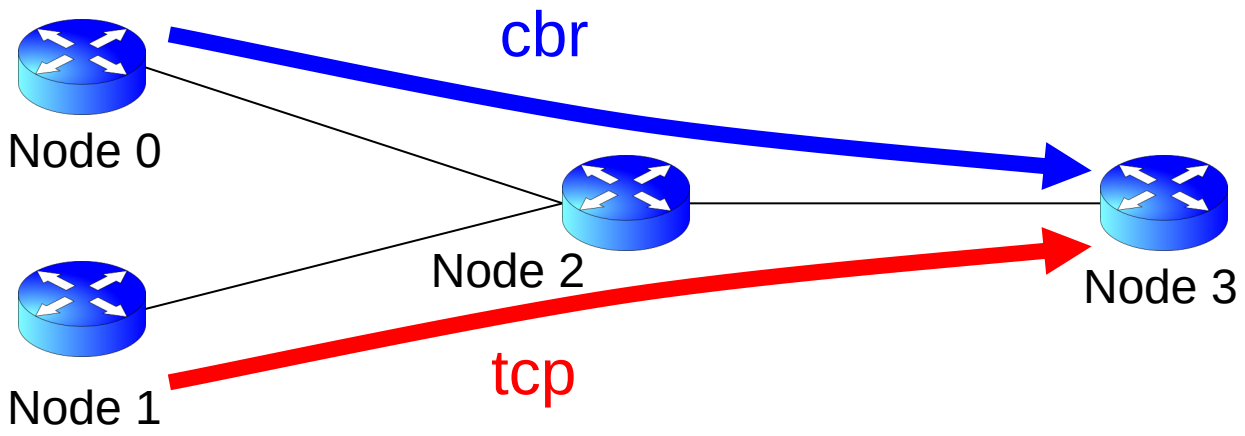
- $1: event
- $2: event occur time
- $3, $4: event occur palace (link)
- $5: kind of packet
- $11: sequence number of packet
- $12: packet ID number

```
+ 1 0 2 cbr 210 ------- 0 0.0 3.0 0 0
- 1 0 2 cbr 210 ------- 0 0.0 3.0 0 0
+ 1.00375 0 2 cbr 210 ------- 0 0.0 3.0 1 1
- 1.00375 0 2 cbr 210 ------- 0 0.0 3.0 1 1
r 1.00556 0 2 cbr 210 ------- 0 0.0 3.0 0 0
+ 1.00556 2 3 cbr 210 ------- 0 0.0 3.0 0 0
- 1.00556 2 3 cbr 210 ------- 0 0.0 3.0 0 0
+ 1.0075 0 2 cbr 210 ------- 0 0.0 3.0 2 2
- 1.0075 0 2 cbr 210 ------- 0 0.0 3.0 2 2
r 1.00931 0 2 cbr 210 ------- 0 0.0 3.0 1 1
                    ⋮
```

Observe Packet Flow

- Look up flow of packet (ID=200, cbr)

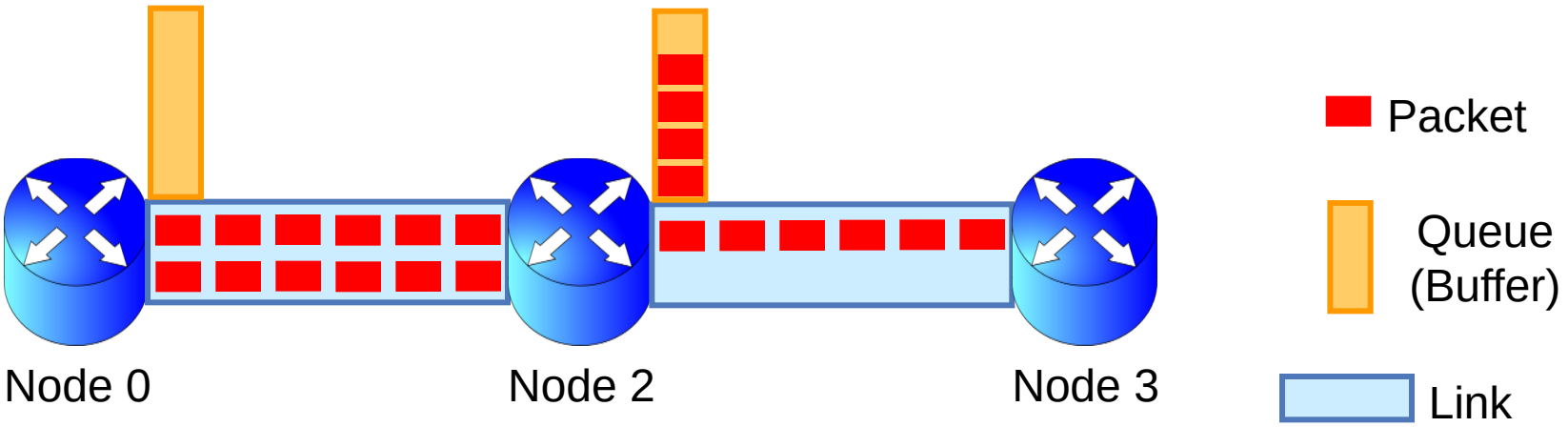
```
% awk '$12 == 200' out.tr
+ 1.64875 0 2 cbr 210 ------- 0 0.0 3.0 173 200
- 1.64875 0 2 cbr 210 ------- 0 0.0 3.0 173 200
r 1.65431 0 2 cbr 210 ------- 0 0.0 3.0 173 200
+ 1.65431 2 3 cbr 210 ------- 0 0.0 3.0 173 200
- 1.672033 2 3 cbr 210 ------- 0 0.0 3.0 173 200
r 1.683153 2 3 cbr 210 ------- 0 0.0 3.0 173 200
```



Flow of Packet

- Observe flow of a packet (ID=200, cbr)

```
% awk '$12 == 200' out.tr
+ 1.64875 0 2 cbr 210 ------- 0 0.0 3.0 173 200
- 1.64875 0 2 cbr 210 ------- 0 0.0 3.0 173 200
r 1.65431 0 2 cbr 210 ------- 0 0.0 3.0 173 200
+ 1.65431 2 3 cbr 210 ------- 0 0.0 3.0 173 200
- 1.672033 2 3 cbr 210 ------- 0 0.0 3.0 173 200
r 1.683153 2 3 cbr 210 ------- 0 0.0 3.0 173 200
```

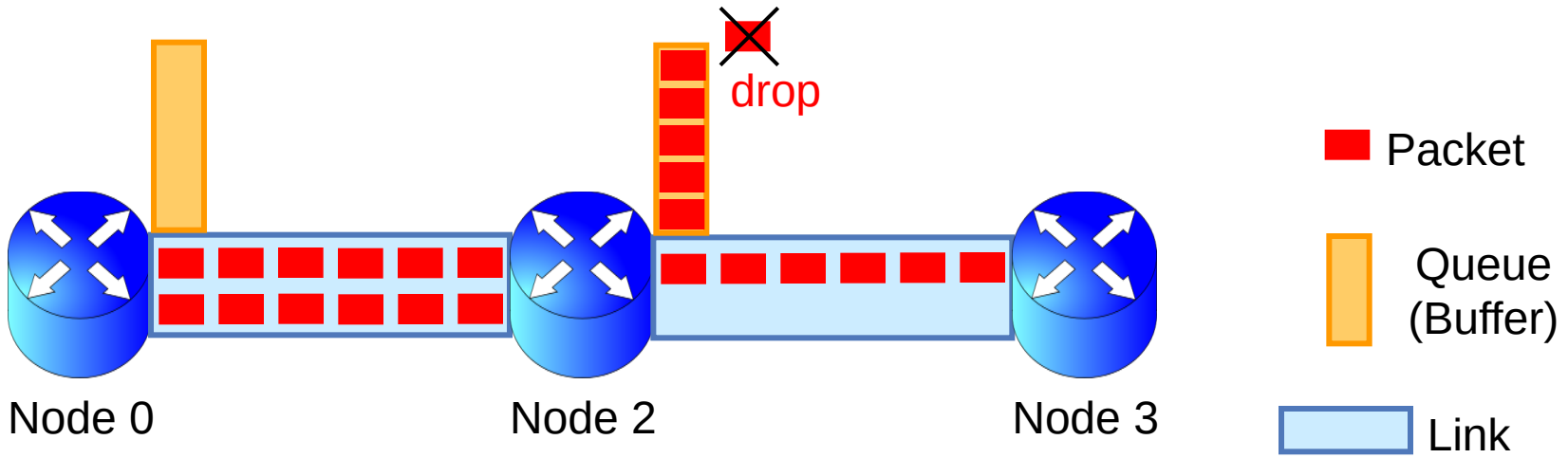


Flow of Packet（Dropped Packet）

- Observer packet (ID=234, cbr)

```
% awk '$12 == 234' out.tr
+ 1.6975 0 2 cbr 210 ------- 0 0.0 3.0 186 234
- 1.6975 0 2 cbr 210 ------- 0 0.0 3.0 186 234
r 1.70306 0 2 cbr 210 ------- 0 0.0 3.0 186 234
+ 1.70306 2 3 cbr 210 ------- 0 0.0 3.0 186 234
d 1.70306 2 3 cbr 210 ------- 0 0.0 3.0 186 234
```

Drop after enter queue



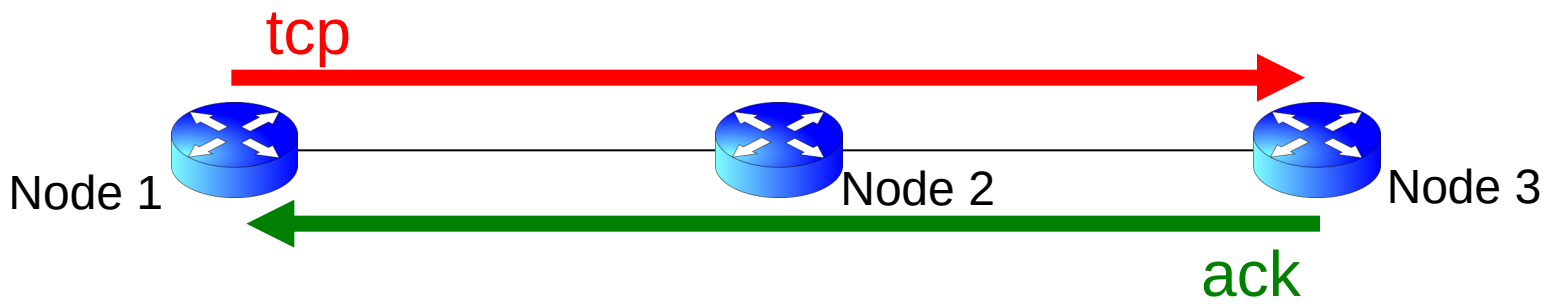
Observe TCP Sequence Number

- Observe (seqno=100, tcp and ack) for a packet

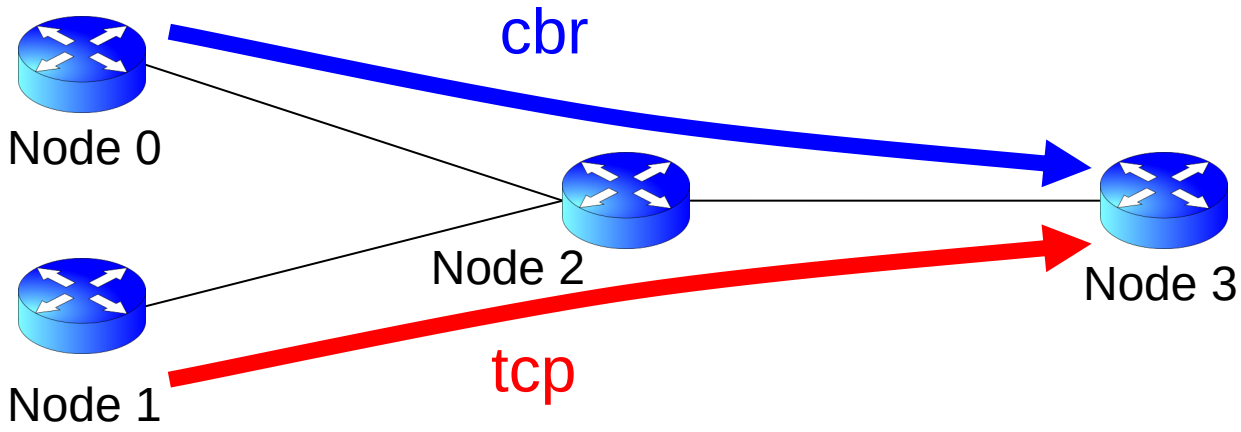
```
% awk '($5=="tcp"||$5=="ack") && $11==100' out.tr
+ 2.489317 1 2 tcp 1040 ------- 1 1.0 3.1 100 631
- 2.49209 1 2 tcp 1040 ------- 1 1.0 3.1 100 631
r 2.499863 1 2 tcp 1040 ------- 1 1.0 3.1 100 631
+ 2.499863 2 3 tcp 1040 ------- 1 1.0 3.1 100 631
- 2.503757 2 3 tcp 1040 ------- 1 1.0 3.1 100 631
r 2.519303 2 3 tcp 1040 ------- 1 1.0 3.1 100 631
+ 2.519303 3 2 ack 40 ------- 1 3.1 1.0 100 643
- 2.519303 3 2 ack 40 ------- 1 3.1 1.0 100 643
r 2.529517 3 2 ack 40 ------- 1 3.1 1.0 100 643
+ 2.529517 2 1 ack 40 ------- 1 3.1 1.0 100 643
- 2.529517 2 1 ack 40 ------- 1 3.1 1.0 100 643
r 2.534623 2 1 ack 40 ------- 1 3.1 1.0 100 643
```

RTT

2.534623
−) 2.489317
0.045306



Extract transmitted packet



- Show the sent cbr packet

```
% awk '$1=="+" && $3==0 && $4==2 && $5=="cbr"' out.tr
```

- Show the received tcp packet

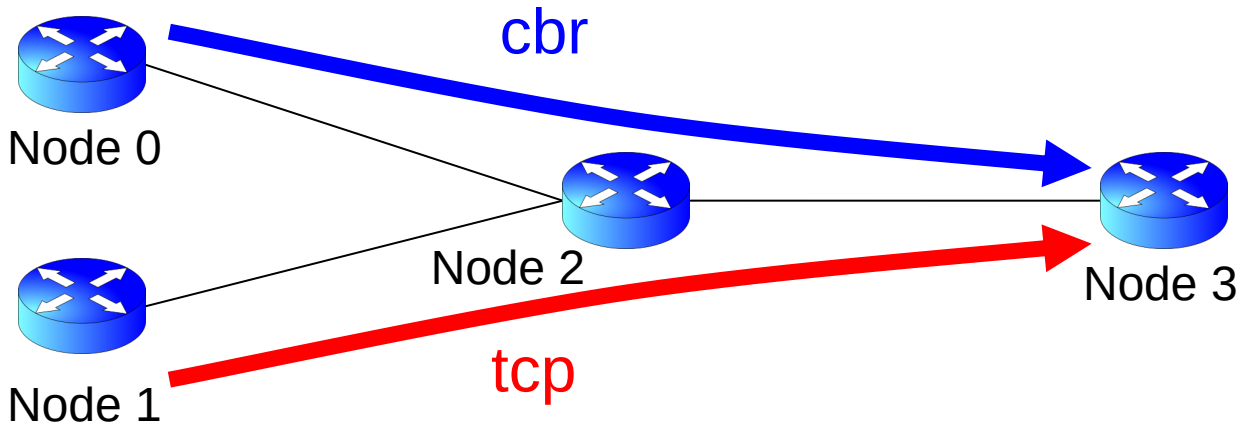
```
% awk '$1=="+" && $3==1 && $4==2 && $5=="tcp"' out.tr
```

- Show the received cbr / tcp packet

```
% awk '$1=="r" && $3==2 && $4==3 && $5=="cbr"' out.tr
```

```
% awk '$1=="r" && $3==2 && $4==3 && $5=="tcp"' out.tr
```

Extract dropped packet



- Show the cbr /tcp drop packet

```
% awk '$1=="d" && $5=="cbr"' out.tr
```

```
% awk '$1=="d" && $5=="tcp"' out.tr
```

※ No consider about place of drop

Count number of transmitted packet and dropped packet

- Count number of sending packet

```
% awk '$1=="+" && $3==0 && $4==2 && $5=="cbr"' out.tr | wc -l
% awk '$1=="+" && $3==1 && $4==2 && $5=="tcp"' out.tr | wc -l
```

- Count number of receiving packet

```
% awk '$1=="r" && $3==2 && $4==3 && $5=="cbr"' out.tr | wc -l
% awk '$1=="r" && $3==2 && $4==3 && $5=="tcp"' out.tr | wc -l
```

- Count number of drop packet

```
% awk '$1=="d" && $5=="cbr"' out.tr | wc -l
% awk '$1=="d" && $5=="tcp"' out.tr | wc -l
```

Evaluate Quality of Transmission

$$\text{Through Put [Mbps]} = \frac{\text{(Total receive packet number)} \times \text{(Packet size)[bytes]}}{\text{(sending time)[s]}}$$

Received amount of data: (Total receive packet number) × (Packet size)[bytes]

Content in header

Time sending from sender: (sending time)[s]

$$\text{Packet drop rate[\%]} = \frac{\text{(total drop packet number)}}{\text{(total sent packet number)}} \times 100$$

No consider about place of drop packet

(total drop packet number)

Packet send from sender: (total sent packet number)

※ M = 10^6
bps = bit/sec
1 bytes = 8 bits

Plot graph using Gnuplot

- Basically command

```
% gnuplot
…
gnuplot> p “data file”
          u x axis data range :y axis data range
          t “legend name”
          w plot style          (※ in one line)
gnuplot> rep “data file” …
…
If the graph is satisfied then
…
gnuplot> set term post color
gnuplot> set output “filename.ps”
gnuplot> rep
gnuplot> q (exit)

% gv filename.ps (confirm result)
% lpr filename.ps (print)
```

Simple gnuplot command

sendai.dat

1 column ($1) : time
2 column ($2) : temperature
3 column ($3) : humid

```
# time temperature humid
     1    18.4      48
     2    16.0      67
     3    16.7      51
     4    16.2      49
     5    16.0      50
     6    15.9      51
     7    16.4      48
     8    15.7      57
     9    16.5      57
    10    16.9      56
    11    16.3      65
    12    16.3      65
                ⋮
    23    13.3      83
    24    12.5      90
```

Data from weather forecast homepage

- axis x is time, axis y is the plot of temperature

```
gnuplot> p “sendai.dat” u 1:2 w lp
```

  - Plot style:l, lp, st, d, i etc
  - Using plot style appropriates with your purpose
  - Chose legend as you desired

- x axis is time, y axis is humid and temperature

```
gnuplot> rep “sendai.dat” u 1:3 w lp
```

- Save as ps (postscript) format

```
gnuplot> set term post color
gnuplot> set output “sendai.ps”
gnuplot> rep
gnuplot> q
```

out.tcp (extract)

- $1: time
- $16: sent sequence number
- $18: congestion window size
- $20: slow start threshold value

```
time: 0.00000  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: -1   hiack: -1   seqno: 0
    cwnd: 1.000  ssthresh: 20  dupacks: 0  rtt: 0.000  srtt: 0.000  rttvar: 12.000 bkoff: 1
time: 1.53096  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 0    hiack: 0    seqno: 1
    cwnd: 2.000  ssthresh: 20  dupacks: 0  rtt: 0.030  srtt: 0.030  rttvar: 0.015  bkoff: 1
time: 1.57005  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 2    hiack: 1    seqno: 3
    cwnd: 3.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
time: 1.57559  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 4    hiack: 2    seqno: 5
    cwnd: 4.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
time: 1.60869  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 6    hiack: 3    seqno: 7
    cwnd: 5.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
time: 1.61535  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 8    hiack: 4    seqno: 9
    cwnd: 6.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
time: 1.62202  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 10   hiack: 5    seqno: 11
    cwnd: 7.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
time: 1.62757  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 12   hiack: 6    seqno: 13
    cwnd: 8.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
time: 1.64733  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 14   hiack: 7    seqno: 15
    cwnd: 9.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
```

⋮

Plot the change of variable by time

- Sequence number ($16: seqno)

```
gnuplot> p “out.tcp” u 2:16 t “seqno” w l
```

- Congestion window size($18: cwnd)
- Slow start threshold value($20: ssthresh)

```
gnuplot> p “out.tcp” u 2:18 t “cwnd” w st
gnuplot> rep “out.tcp” u 2:20 t “ssthresh” w st
```

- The minor between congestion window size and 20

```
gnuplot> p “out.tcp” u 2:($18 < 20 ? $18 : 20) w st
```

gnuplot – supplement

- Assign label

```
gnuplot> set xl [x axis label]
gnuplot> set yl [y axis label]
gnuplot> rep
```

- Specific display range

```
gnuplot> set xr [x axis minimum value: maximum value]
gnuplot> set xr [x axis minimum value: maximum value]
gnuplot> rep
```

- Single logarithmic chart
  - Single logarithmic chart of X axis

```
gnuplot> set log x
gnuplot> rep
```

Supplement for report

- Execute simulation, analysis result
  - It is possible to use other language
    - C language
    - Perl, Ruby, Java Script, …
- Draw graph
  - It is possible to use other application
    - Hand writing on paper
    - Excel, OpenOffice Calc, …
- Report
  - By paper report
    - Hand writing
    - LaTeX
    - Word, OpenOffice Writer, …