Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED

Kaixin Xu
Mario Gerla
Lantao Qi
Yantai Shu

MobiComm 2003

Prenseted by Ronak Bhuta

Date : October 9, 2007

Overview

- Introduction - Challenges Posed to TCP design in Wireless Ad Hoc Networks and Prior work
- RED – Its simulation over Ad Hoc network and reasons for it to not work
- Neighborhood and its Distributed queue
- Neighborhood Random Early Detection (NRED) – NCD, NCN and DNPD
- Verification and Parameter Tuning
- Performance Evaluation of NRED
- Discussion
- Conclusion and Comments

Challenges Posed to TCP design in Wireless Ad Hoc Networks

- Topology changes and path changes cause TCP to go into exponential backoff
- 2nd problem is the critical significance of the congestion window size in use
- Significant TCP unfairness being the 3rd problem
- This paper focuses on TCP fairness in ad hoc networks

Prior work

- Paper attacks the unfairness problem at the network layer
- It explores the relation between TCP unfairness and early network congestions
- RED can improve congestion control and fairness in wired networks

RED

- RED monitors average queue size at each buffer
- It drops/marks packets with a drop probability, if queue size exceeds a predefined threshold
- Drop probability is calculated as a function of average queue size
- It improves congestion control and fairness by dropping packets proportional to connections bandwidth share

Simulation environment used for experiments

- Simulation platform used is QualNet simulator
- Channel bandwidth is 2Mbps
- IEEE 802.11 MAC DCF
- TCP NewReno used with maximum Segment Size set to 512 bytes
- Buffer size at each node is 66 packets
- Static Routing

TCP unfairness And RED in Ad Hoc Networks

- FTP 2 is starved as RED does not improve fairness but improves throughput

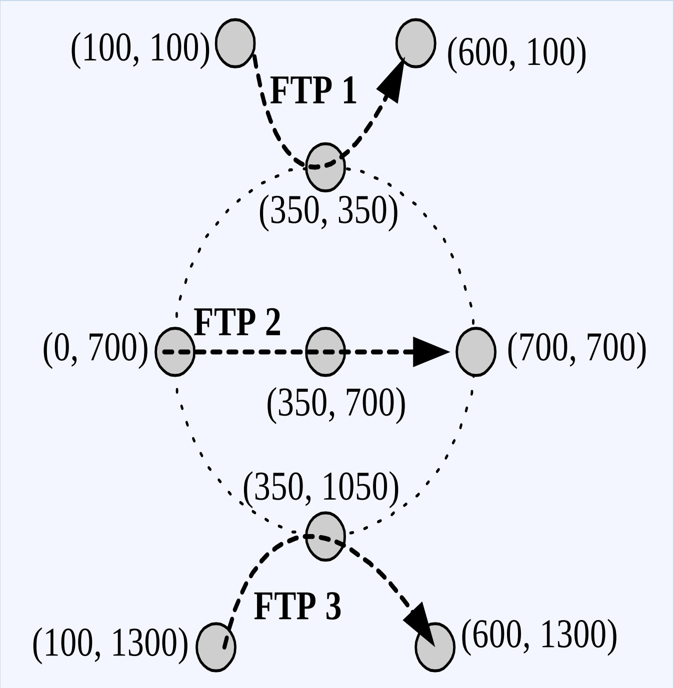


Figure 1: A Wireless Specific scenario for testing TCP unfairness with RED

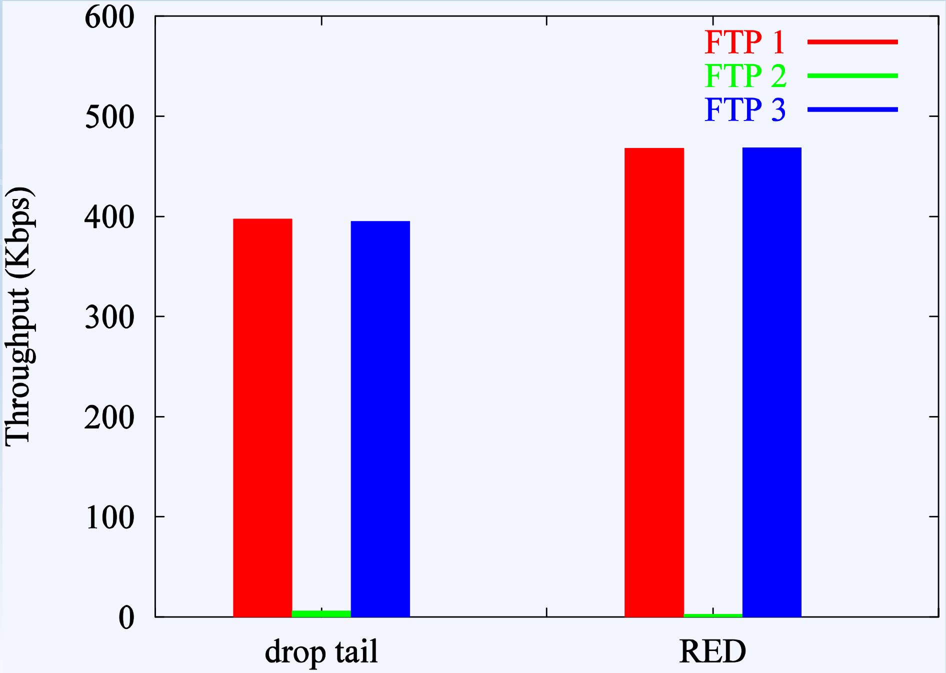


Figure 2: Overall throughput of flows at the end of simulation with RED's max_p equal to 0.06

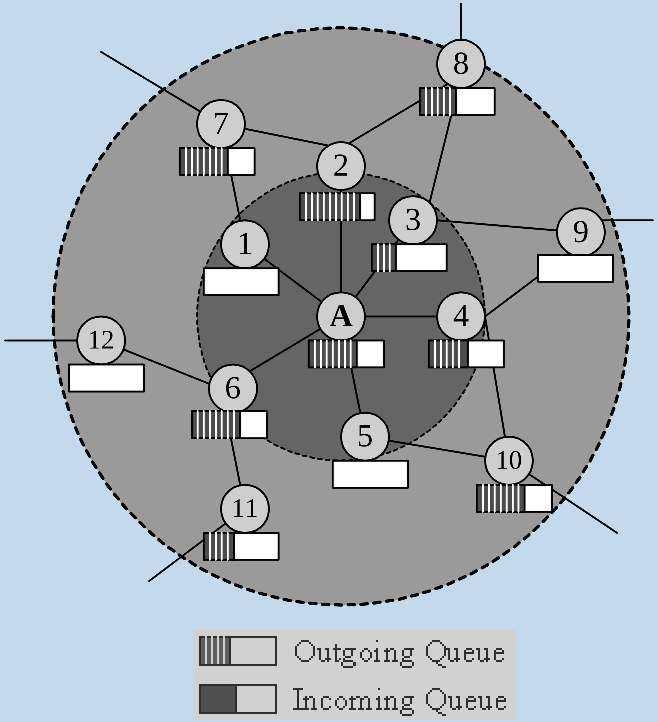
Reference: http://www.cacs.louisiana.edu/~wu/619/presentations/NRED.ppt

Why RED does not Work?

- Penalized TCP flows may experience queue build up
- Multiple node contribute to congestion
- Unfairness is caused as nodes drop packets unaware of their or others', bandwidth share and contribution to congestion
- Queue at any single node cannot reflect the network congestion state
- Extend RED to entire congested area – Neighborhood of node

Neighborhood and its Distributed queue

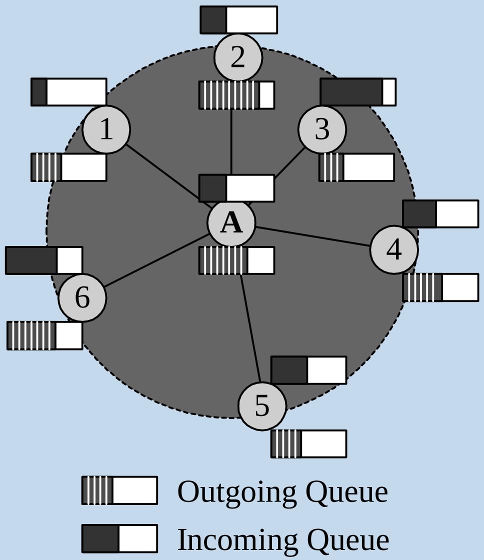
- A node's neighborhood consists of the node itself and the nodes which can interfere with this node's signals
- 1- hop neighbors' directly interferes and 2 – hop nodes may interfere
- Queue size of a neighborhood reflects the degree of local network congestion



Reference: http://www.cacs.louisiana.edu/~wu/619/presentations/NRED.ppt

Simplified Neighborhood Queue Model

- Simplified neighborhood includes only 1-hop neighbors
- 2–hop neighbors have a lot of communication overheads so only those packets of 2-hop that are directed towards 1-hop are included
- Each node has 2 queues- incoming and outgoing queue
- Distributed Neighborhood queue- the aggregate of these local queues



Reference: http://www.cacs.louisiana.edu/~wu/619/presentations/NRED.ppt

Characteristics of distributed Neighborhood Queue

- Consists of multiple queues located at the neighboring nodes
- Queue is not a FIFO queue due to **location dependency?**
- Priority of sub-queues change dynamically depending on topology changes/ traffic pattern changes
- TCP flows sharing the same neighborhood may get different feedbacks in terms of packet delay and loss rate

Neighborhood Random Early Detection (NRED)

- RED extended to the distributed neighborhood queue
- Key problems
  - Computing average queue size of distributed neighborhood queue
  - Spreading congestion notification amongst neighbors
  - Calculating proper drop probability at each node
- Components of Neighborhood RED tackling above problems
  - Neighborhood Congestion Detection (NCD)
  - Neighborhood Congestion Notification (NCN)
  - Distributed Neighborhood Packet Drop (DNPD)

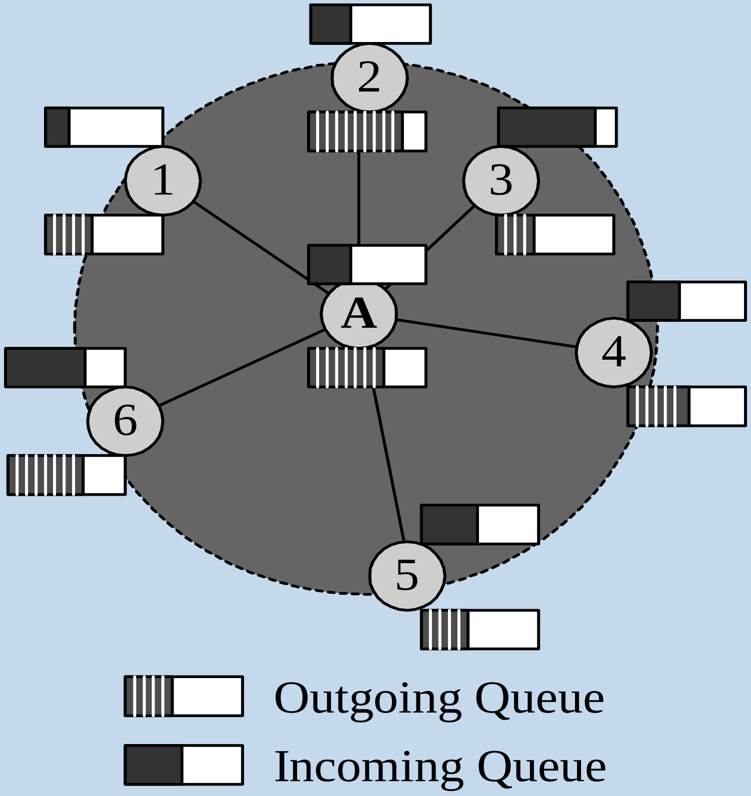
Neighborhood Congestion Detection

- Direct way: Announce queue size upon changes
  - Too much overhead worsening the congestion
- Method proposed in the paper: Indirectly estimating an index of queue size by monitoring wireless channel utilization
  - Channel Utilization ratio $U_{busy} = \frac{T_{interval} - T_{idle}}{T_{interval}}$
  - Queue size index $q = \frac{U_{busy} * W}{C}$

    w is channel bandwidth, *c* is a constant packet size
- Average queue size is calculated using RED's algorithm
- Congestion: queue size exceeds minimum threshold

Neighborhood Congestion Detection

- A node will monitor 5 different radio state
  - Transmitting T_{tx}
  - Receiving T_{rx}
  - Carrier sensing busy T_{cs}
  - Virtual carrier sensing busy T_{vcs}
  - Idle T_{idle}
- State 1&2 is for current node, 3&4 is for its neighbors. The authors assume state 5 means empty queue.
- When a packet in any outgoing queue is transmitted, node A will detect the medium as busy.
- When a packet is received to any incoming queue, node A can also learn this through the CTS packet.



Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)

Neighborhood Congestion Detection

$$(1)\ U_{busy} = \frac{T_{interval} - T_{idle}}{T_{interval}};\ (2)\ U_{tx} = \frac{T_{tx}}{T_{interval}};\ (3)\ U_{rx} = \frac{T_{rx}}{T_{interval}};$$

$$T_{interval} = T_{tx} + T_{rx} + T_{cs} + T_{vcs} + T_{idle}$$

Assume W is channel bandwidth and the average packet size is C bits

$$q = \frac{U_{busy} * W}{C}; avg = (1 - w_q) * avg + w_q * q$$

We can use the same way to calculate avg_{tx} *and* avg_{rx}

Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)

Neighborhood Congestion Notification

- Under NRED, a node checks the estimated average queue size *avg* periodically and compares it with old *min* threshold. The node calculates a drop prob p_b and broadcasts it to its neighbors if the following Constraints Holds for the current nodes.
  - The calculated Pb is larger than 0.
  - Current node is on the path of one or more flows
  - Node is suffering in channel contention (by comparing $avg_{tx} + avg_{rx}$ with a threshold)
  - Didn't receive any NCN in the past interval with a larger $normalizedP_b$. Otherwise the neighborhood is more congested.
- NCN packet field includes <*packetType*, $normalizedP_b$,*lifeTime*>

Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)

Neighborhood Congestion Notification

Algorithm 5.1: CALCULATEPB()

comment: Procedure to calculate Drop Probability p_b

Saved Variables:

avg: *average queue size*

Fixed Parameters:

min_{th}: *minimum threshold for queue*

max_{th}: *maximum threshold for queue*

max_p: *maximum value for* p_b

T_{NCN}: *time interval for performing this action*

for each $\mathcal{T}_{NCN}$

$avg \leftarrow estimatedQueueSize()$

if $min_{th} \leq avg < max_{th}$

$p_b \leftarrow max_p * (avg - min_{th})/(max_{th} - min_{th})$

$normalizeP_b \leftarrow p_b/avg$

else if $max_{th} \leq avg$

$p_b \leftarrow 1$

$normalizedP_b \leftarrow 1$

Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)

Distributed Neighborhood Packet Drop

- When a node received a NCN with a non zero *normalized*P_b, the local drop prob p_b is caculated as *normalized*P_b* $(avg_{tx} + avg_{rx})$

Algorithm 5.2: RANDOMDROP()

comment: Actions performed at the outgoing queue

Saved Variables:

$count_{tx}$*: outgoing pkts arrived since last drop*

avg_{tx}*: average outgoing queue size*

Other Parameters:

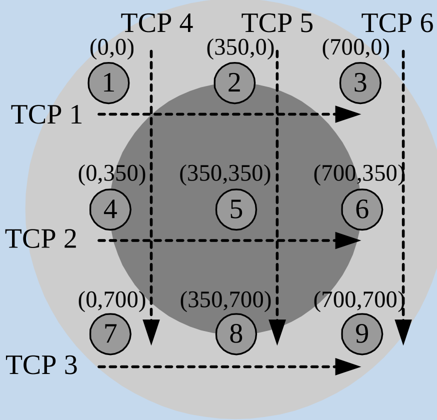
p_a*: current packet dropping probability*

for each *packet arrival*
  $count_{tx} \leftarrow count_{tx} + 1$
  if $normalizedP_b < 1$
    $p_b \leftarrow normalizedP_b * avg_{tx}$
    $p_a \leftarrow p_b / (1 - count_{tx} * p_b)$
  else $p_a \leftarrow 1$
  if $p_a > 0$
    $aRandomNumber \leftarrow random([0, 1])$
    if $aRandomNumber \leq p_a$
      drop the arriving pkt
      $count_{tx} \leftarrow 0$
  else $count_{tx} \leftarrow -1$

Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)

Verification of Queue Size Estimation

- It estimates channel utilization as an approximation for neighborhood queue
- Estimating Node5's neighborhood queue size index
- Gets real queue size by recording queue size at individual nodes
- Evaluated NRED for frequent queue size changes by replacing FTP flow with HTTP flows
- Parameters $T_{interval}$=*100ms* and w_q=*0.2*



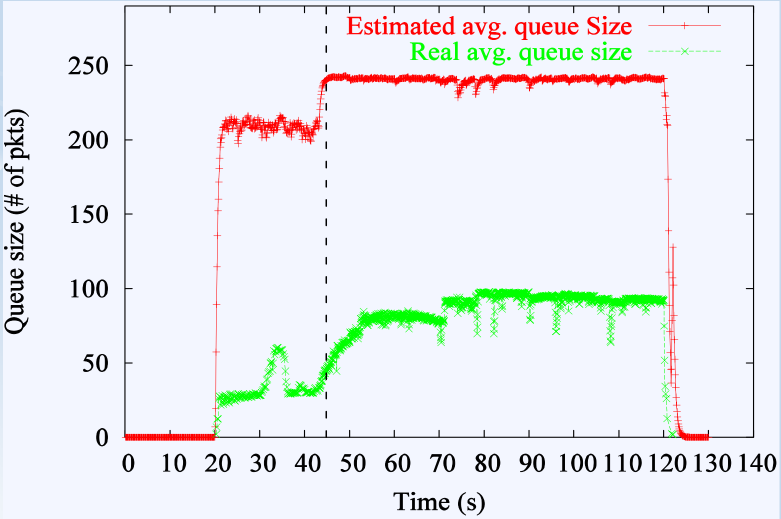


Figure 6: Shows FTP/TCP connections

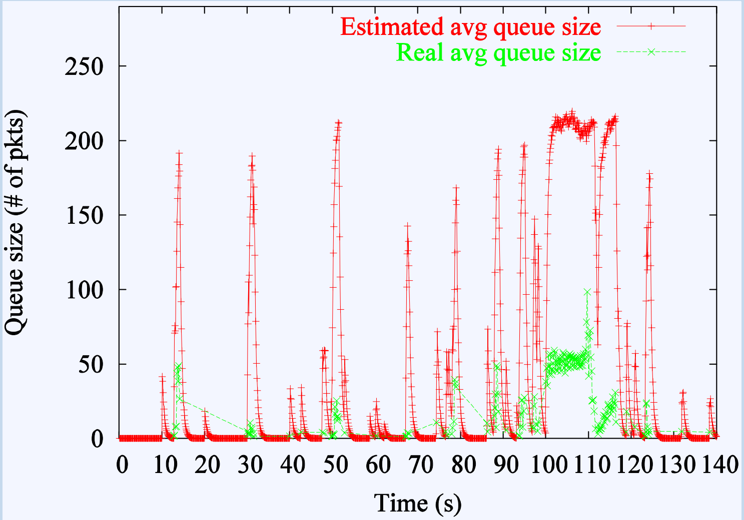


Figure 7: Shows HTTP/TCP connections

Reference: http://www.cacs.louisiana.edu/~wu/619/presentations/NRED.ppt

Parameter Tuning

- Parameter Tuning with Basic Scenarios with hidden and exposed terminal scenario
  - Hidden Terminal
    - A hidden node is one that is within the interfering range of the intended destination but out of the sensing range of the sender, which can cause collisions on data transmission
  - Exposed Terminal
    - An exposed node is one that is within the sensing range of the sender but out of the interfering range of the destination

Reference: http://ail.tju.edu.cn/report/kt-liuyan.ppt#256,1,Investigation on Neighborhood in Ad Hoc Wireless Network (Ad Hoc网络中邻居内若干问题的研究)

Parameter Tuning with Basic Scenarios

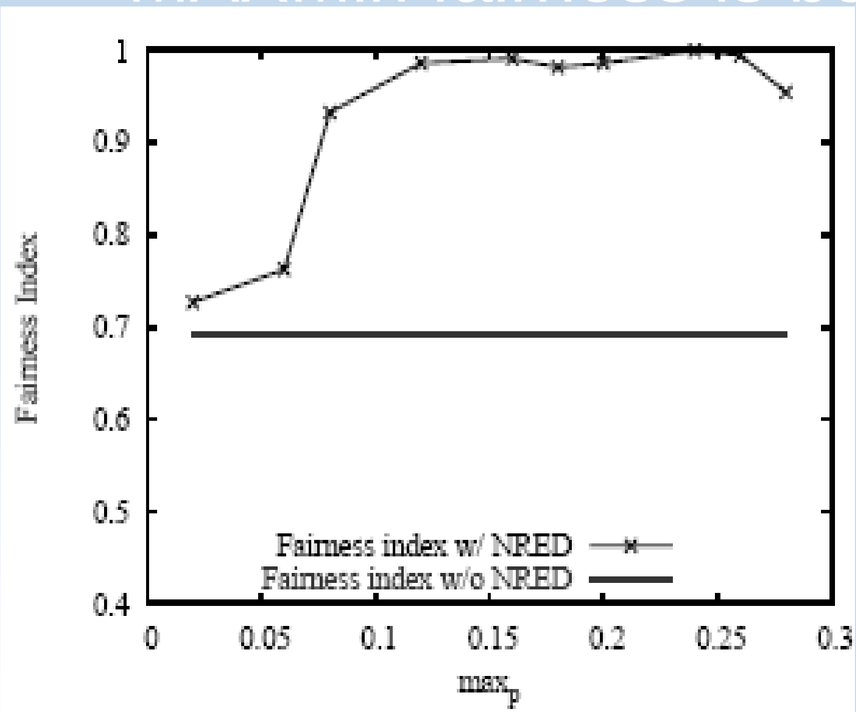
Figure 8: The hidden terminal scenario, where Node 2 is hidden by transmission from node 4 to node 3 and Node 3 is hidden by transmission from node 1 to node 2.

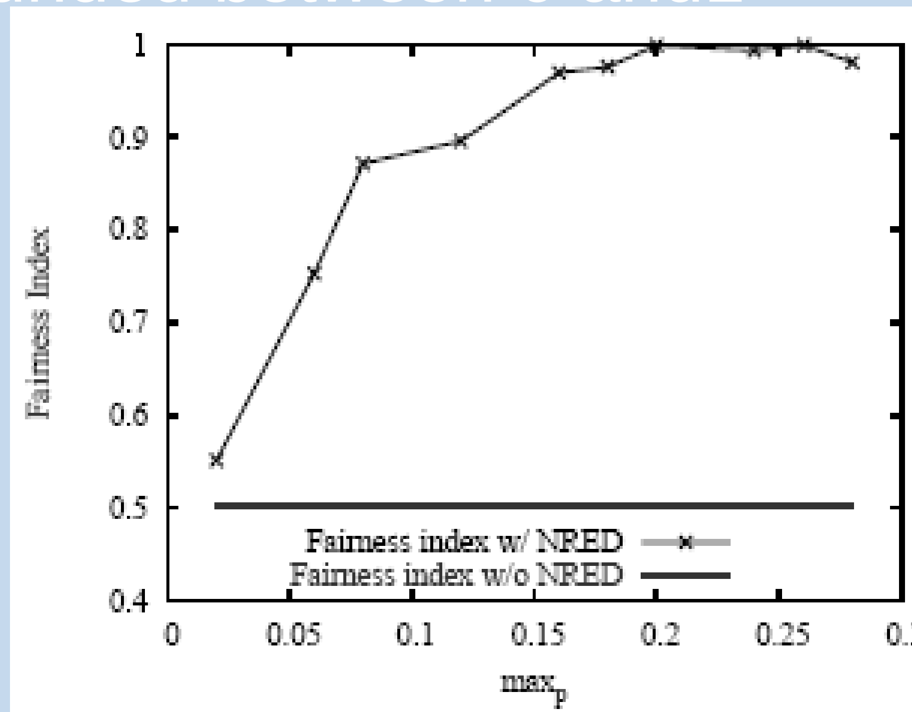
Figure 9: The exposed terminal scenario, where node 2 is exposed to transmissions from node 3 to node 4.

Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)

Parameter Tuning with Basic Scenarios

- Fairness index $F(X_1, X_2) = \frac{(X_1 + X_2)^2}{2(X_1^2 + X_2^2)}$ under hidden and exposed terminal scenario
- MAXMin fairness is bounded between 0 and1

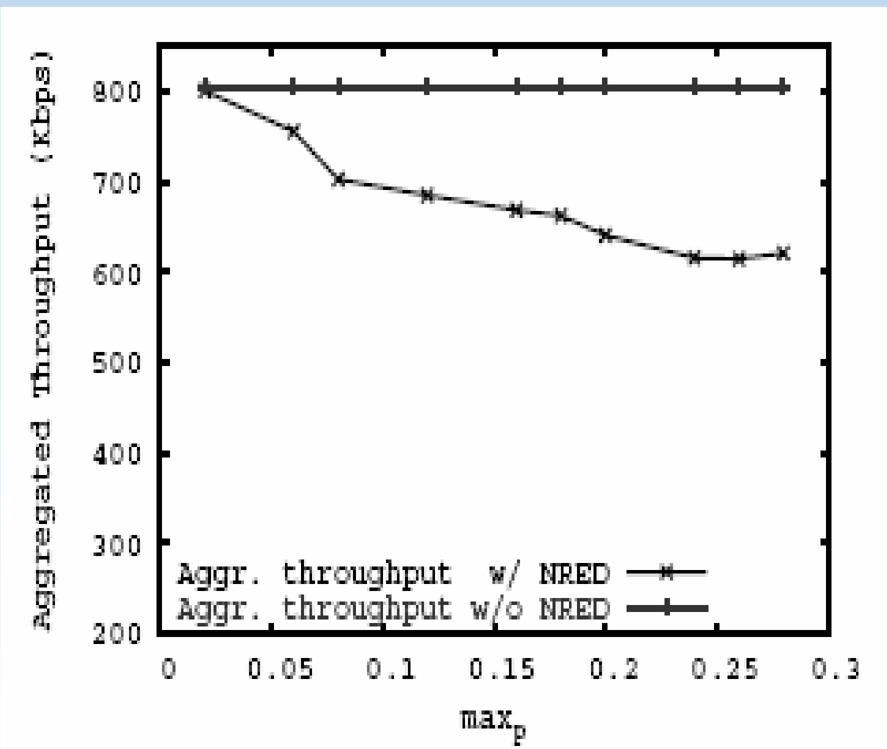


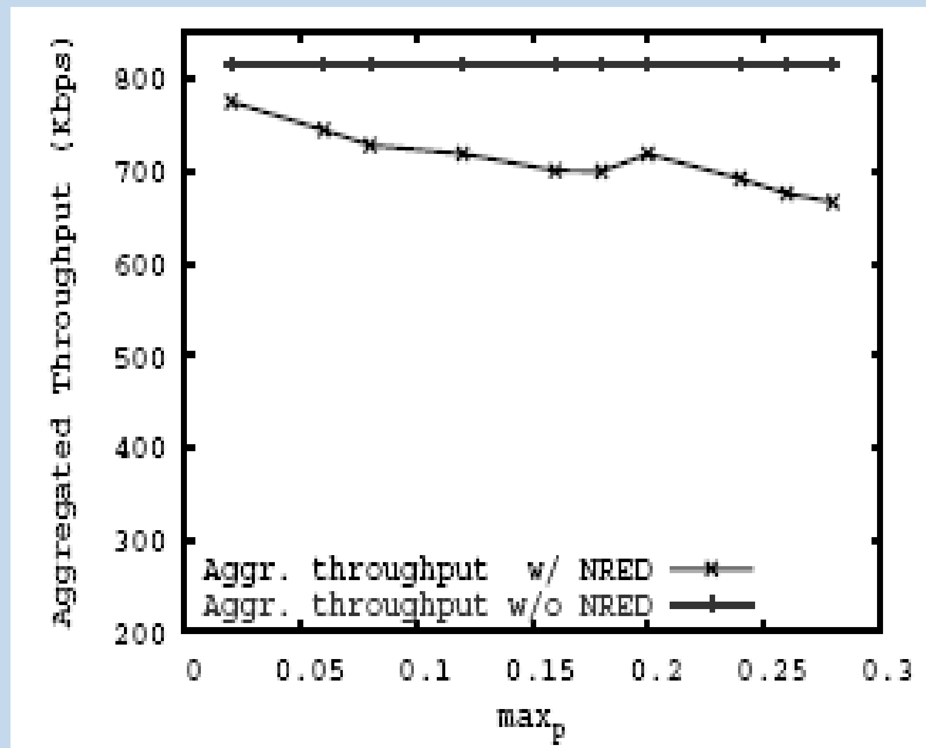


Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)sv

Parameter Tuning with Basic Scenarios

- Aggregated Throughput (kbps) under hidden and exposed terminal situation

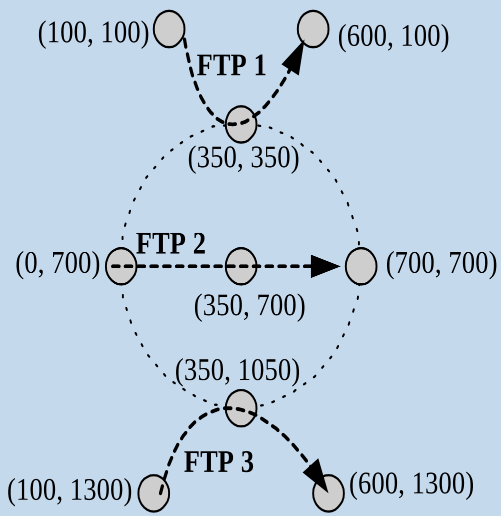


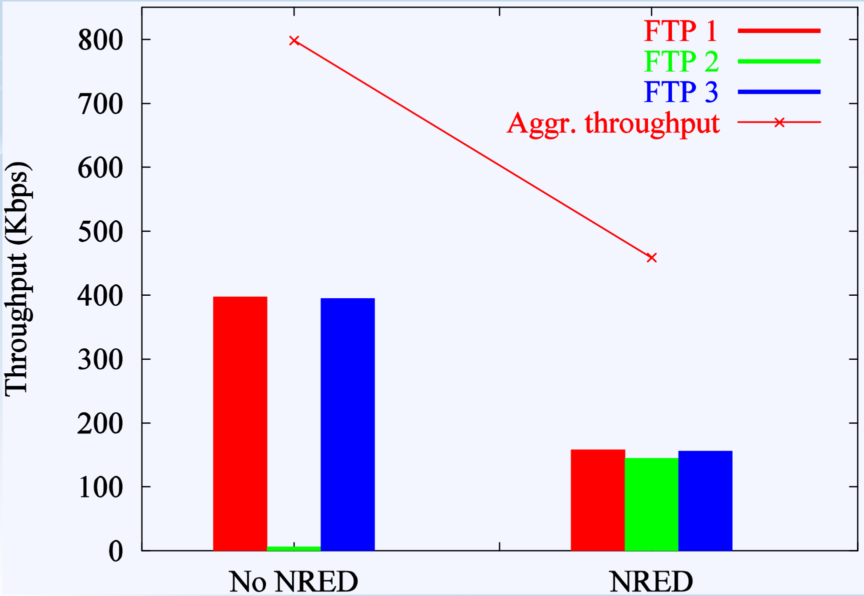


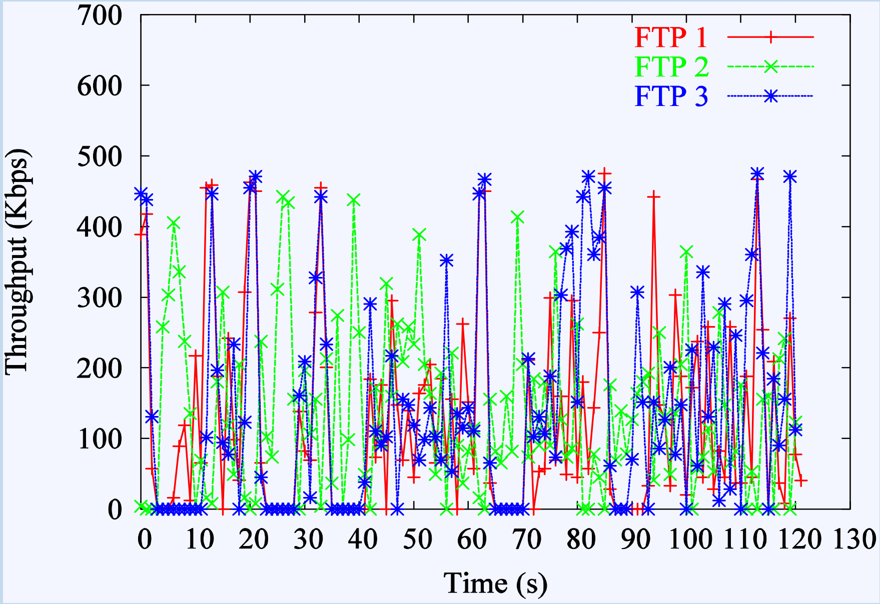
Reference: http://vorlon.case.edu/~sxj63/EECS600-S05/Lecture0413.ppt#264,10,Neighborhood Congestion Detection (NCN)sv

Performance Evaluation: Simple Scenario

- Both long-term and short-term fairness is achieved
- Loss of aggregated throughput
  - There is a Tradeoff between fairness and throughput
  - Channel is slightly not fully utilized



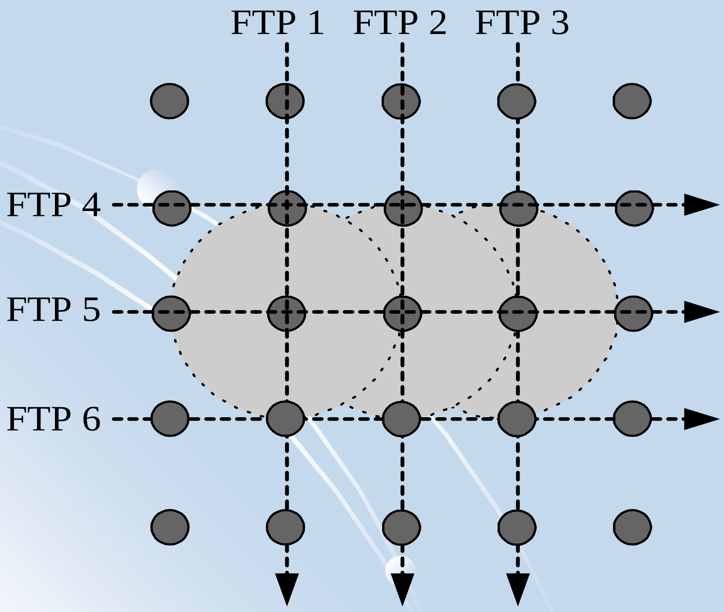


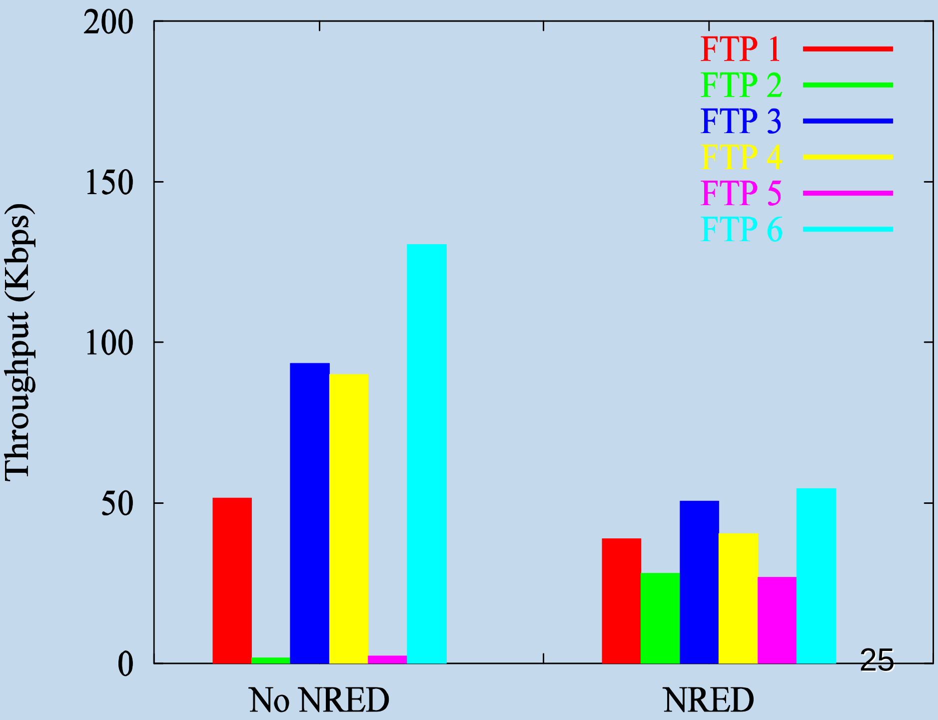


Reference: http://www.cacs.louisiana.edu/~wu/619/presentations/NRED.ppt

Performance Evaluation: Multiple Congested Neighborhood

- Multiple congested neighborhoods
- FTP2 & FTP 5 have more competing flows, are easy to be starved

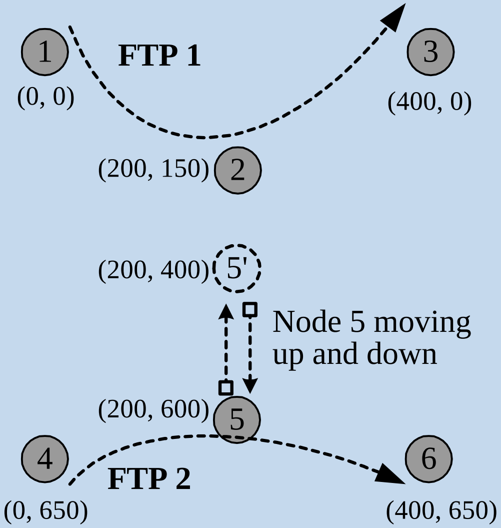


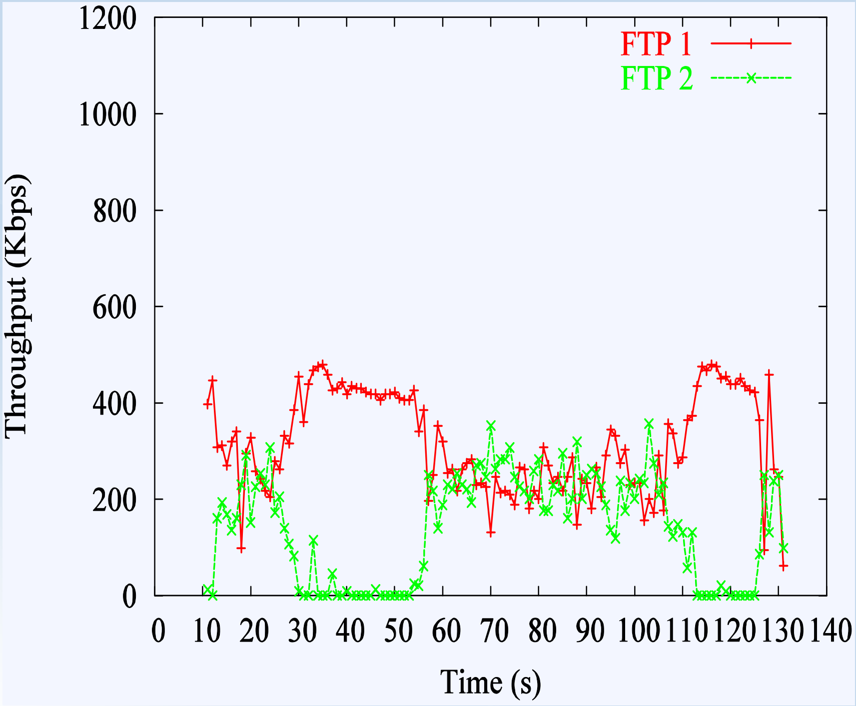


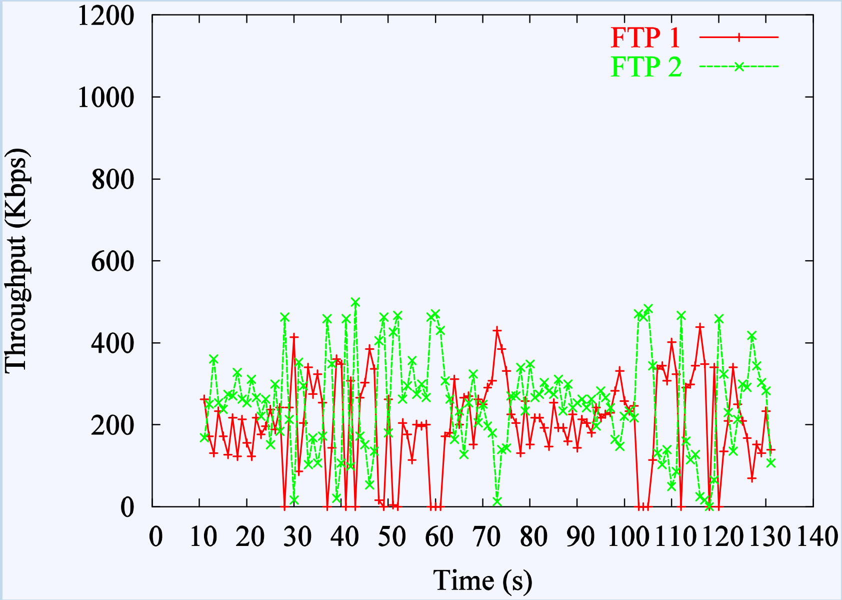
Reference: http://www.cacs.louisiana.edu/~wu/619/presentations/NRED.ppt

Performance Evaluation: Mobility

- Node 5 moves up and down
  - Moving Up: two flow interfere with each
  - Moving down: No much interference
- NRED can adapt to mobility

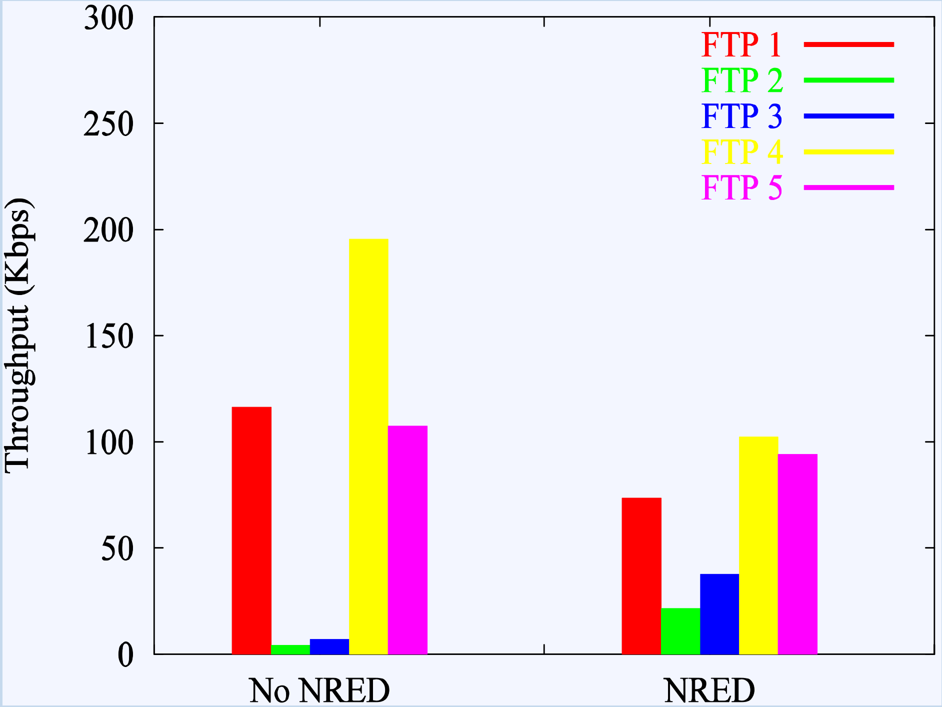






Performance Evaluation: Realistic Scenario

- 50 nodes randomly deployed in 1000mX1000m field
- 5 FTP/TCP connections are randomly selected
- No mobility



Discussion

- Significant TCP unfairness has been found and reported in ad hoc networks
- NRED is a network layer solution
  - Easy to implement
  - Incremental Deployment
- Major Contribution
  - Model of neighborhood queue
    - Distributed neighborhood queue
    - Not FIFO
  - Network layer solution for enhancing TCP fairness in Ad Hoc networks

Discussion (contd)

- Random mobility may reduce aggregate throughput by erroneous invoking of congestion control scheme
- Unlike flow based fair scheduling algorithms, does not require topology information thus has low overhead
- TCP flows are randomly dropped at congested neighborhood which is not efficient for network throughput because the packets have already consumed some bandwidth before reaching the congested area
  - suggested remedy- explicit congestion notification using ECN bit
- Not effective for short-lived TCP connections

Conclusion

- By Detecting congestion and dropping packets proportionally to a flow's channel bandwidth usage, the NRED is able to improve TCP fairness.

- The major contributions of this work are the concept of a distributed neighborhood queue and the design does not require MAC modification.

Comments

- The estimated queue size does not reflect future increase of the queue size after the real average queue size exceeds a certain threshold
- NRED not evaluated for Dynamic Routing and Random Mobility
- Need to study the performance with different MAC protocols

Thank you