Experimental Performance Evaluation of AODV Implementations

Koojana Kuladinithi, Asanga Udugama, Niko Fikouras, Carmelita Görg
University of Bremen
Communication Networks (ComNets)
Otto-Hahn-Allee NW 1
28359 Bremen
{koo/adu/niko/cg}@comnets.uni-bremen.de

Overview

- Motivation
- Implementation Considerations
- Testbed Set UP
- UDP Results Analysis
- TCP Results Analysis
- Further Investigations
- Conclusion

Motivation

- Lack of performance evaluation in real AODV networks
  - Simulation
    (Easy to develop, Simple to get/analyze/repeat Results)
  - Implementation
    (Difficult to develop, code should be deployable, results should be realistic)

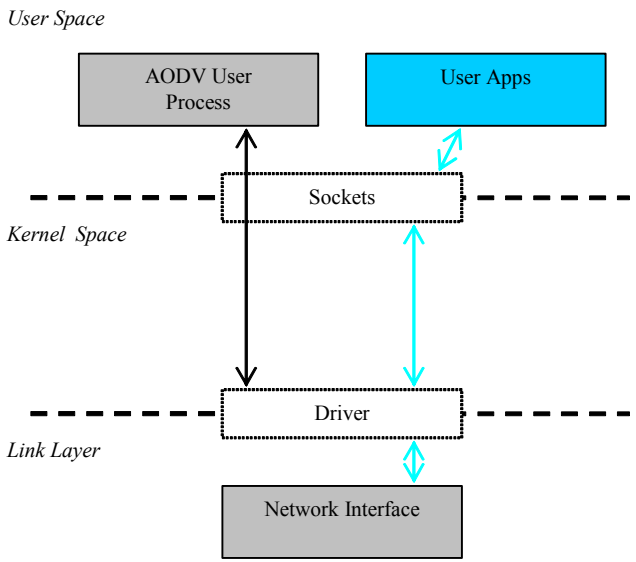
Implementation Considerations

- Unicast and broadcast/multicast communications
  - RREQ, RERR – Multicast/Broadcast
  - RREP, RREP-ACK, RERR – unicast
- Intercepting IP packets that require a route
  (to commence the Route Discovery process)
- Information of IP packets that utilize the existing routes
  (to update route lifetime)
- Manipulation of the Routing Table (Add, Delete & Modify routes)
- Support for a Timer Mechanism
  (Route discovery process, Route maintenance,
  Route deletion, Hello Messages, etc.)
- Packet Queuing (Buffering)
- Link break detection (Hello Messages, Link Layer)

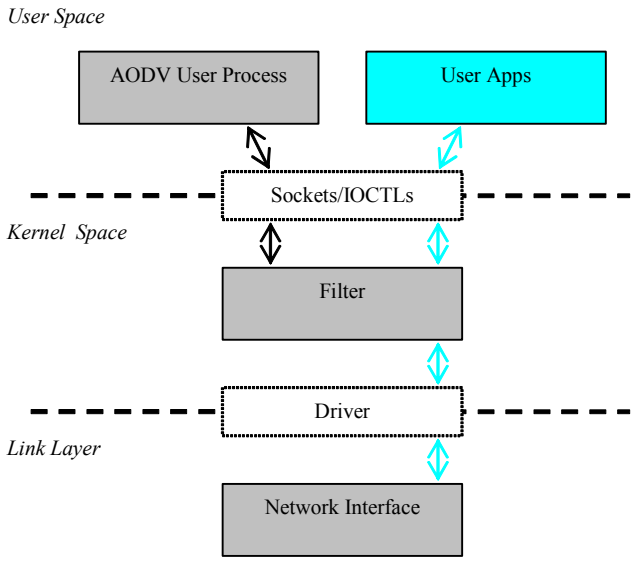
Implementation Considerations , cont.

- Possible Architecture Models
  - *Interface Monitoring Architecture (Purely on User Space)*
  - *Netfilter based Architecture (User Space + Kernel Space)*
  - *Kernel based Architecture (Kernel Space)*

Implementation Considerations , cont.

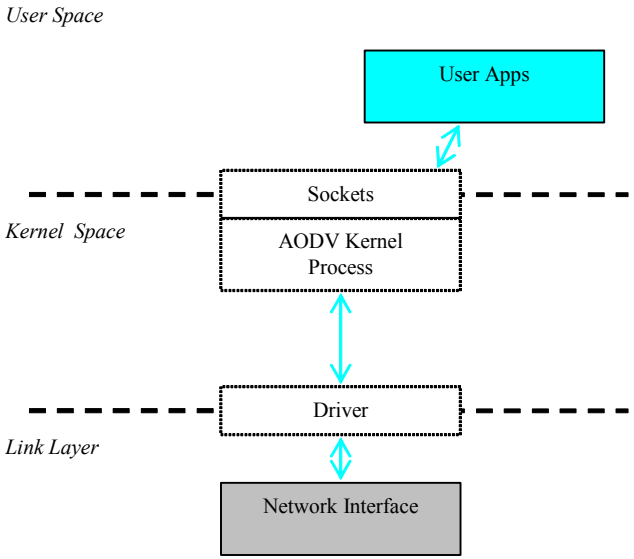


Interface Monitoring Architecture (Purely on User Space)



Filter based Architecture (User Space + Kernel Space)

Implementation Considerations , cont.

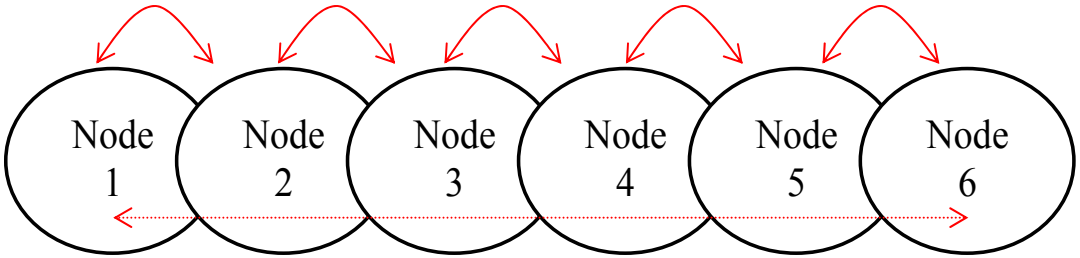


Kernel based Architecture (Kernel Space)

Comparision of Implementations

- UU-AODV implementation by Uppsala University, Sweden
  - Filter based architecture on Linux platform
  - Notebooks & PDAs
  - Version 0.7
- JAdhoc- AODV implementation by University of Bremen, Germany
  - Interface monitoring architecture for Linux/Windows Platforms
  - Notebooks & PDAs
  - Version 0.2

Testbed Set-Up



- Static (non-mobile) Ad-hoc n/w (conference room, lecture room)
- Same frequency CH
- MACKill Filter (at the n/w layer)
- 3 modes of configurations
  - "UU (version 0.7)" - AODV implementation by Uppsala University, Sweden
  - "JAdhoc (version 0.2)" - AODV implementation by University of Bremen, Germany
  - "Static" – Set routes manually in ad-hoc network, (No AODV process is running)

Testbed Set UP, Cont..

- Hardware
  - 6 Sony Vaio notebooks
    (Mobile AMD Duron processors running at 1.1 GHz and 256 MB of RAM)
  - For wireless connectivity, Cisco Aironet IEEE 802.11b wireless cards
    (set in ad-hoc mode on channel 1, no WEP encryption)
- Software
  - Linux Mandrake 8.2 distributions and the Linux Kernel 2.4.19
  - In order to perform the experimental evaluation while maintaining the notebooks in close physical distance to each other the MACKill software was installed on each notebook to filter IP Packets at the link layer
  - Iperf traffic generator software to generate UDP and TCP flows
  - Web100 to measure a selected set of TCP parameters (RTO, CWND)
  - TCPDUMP to identify DupAcks
  - The WLAN interface of each notebook has been allocated a different IP address from a different sub network

Testbed Set-Up - UDP Measurements

- Found the maximum data rate that can be used without any packet loss at the link layer
  - Use Static Mode with 2 nodes
  - Max of 3.6 Mbps of UDP stream without any packet loss

[When using 6 nodes, ~ 0.5Mbps UDP stream can be used to send data without any link layer packet loss]

- Uni-directional data flow at 0.5Mbps is used to evaluate upper layer performance
  - Actual Load
  - Packet Delivery Fraction
  - Out of order Packets

Testbed Set Up – TCP Measurements

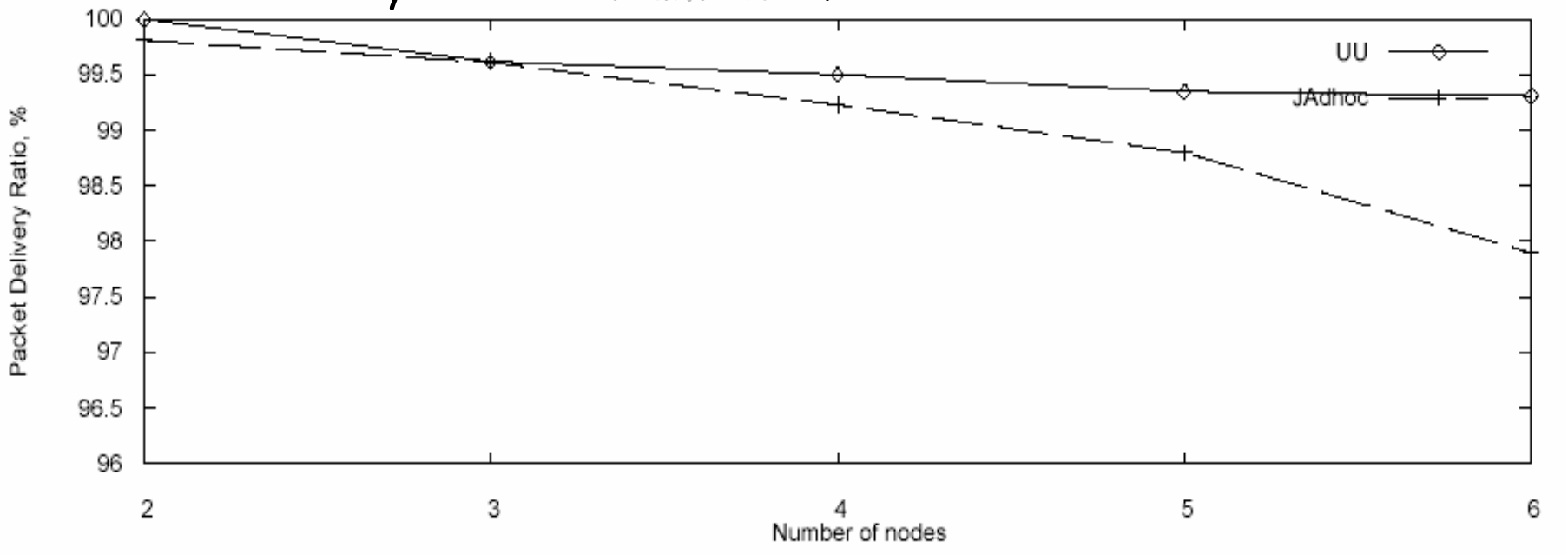
- TCP throughput is measured
- Throughput variations are analysed using the following parameters:
  - RTO
  - CWND
  - DupAcks

- Both UDP & TCP measurements are taken
  - With the increase of nodes from 2 to 6(From 1 – 5 hops), while taking node 1 as the sender
  - For all 3 modes (Static, UU, JAdhoc)
  - For the duration of 60 seconds
- All UDP measurements and TCP throughput are taken at the receiver side, except TCP parameters

UDP Results Analysis

UDP Results Analysis

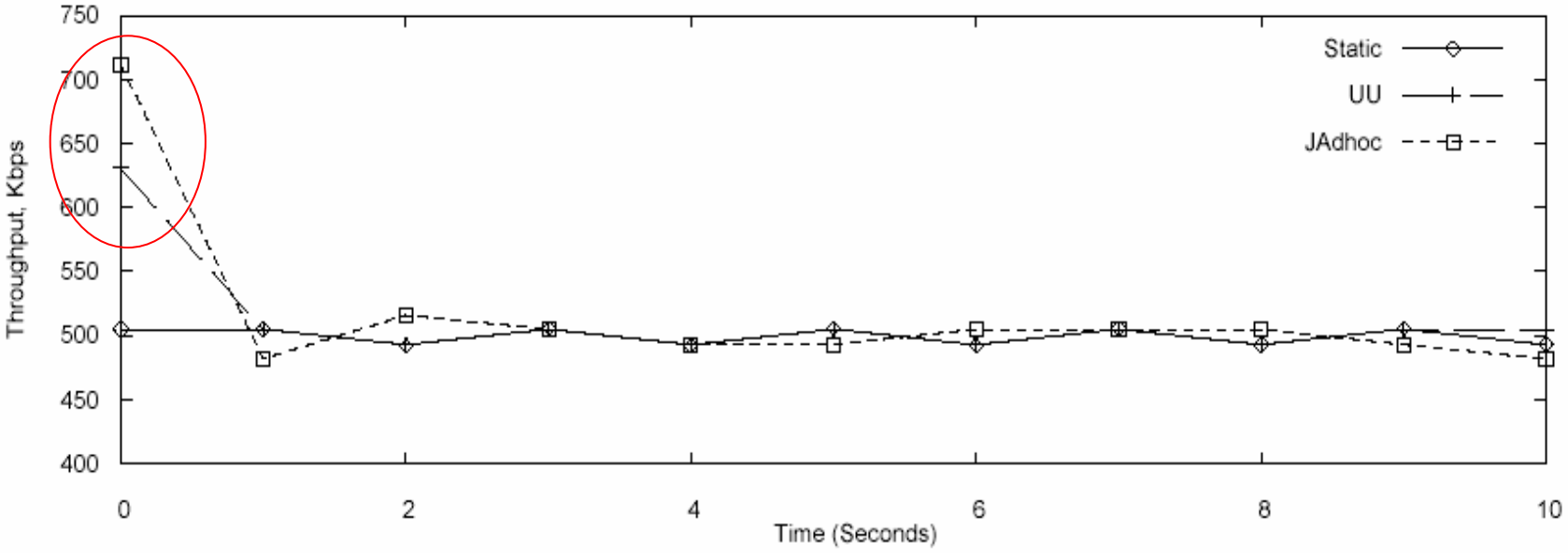
Packet Delivery Ratio vs Number of nodes



- Less with the increase of nodes (hop counts)
- Lies between 98 – 100 %
- Most packet loss ocuurs when releasing buffered packets (just after the route discovery time – explained in the next slide)
  - Therefore, PDR remains close to 100% for longer UDP sessions

UDP Results Analysis

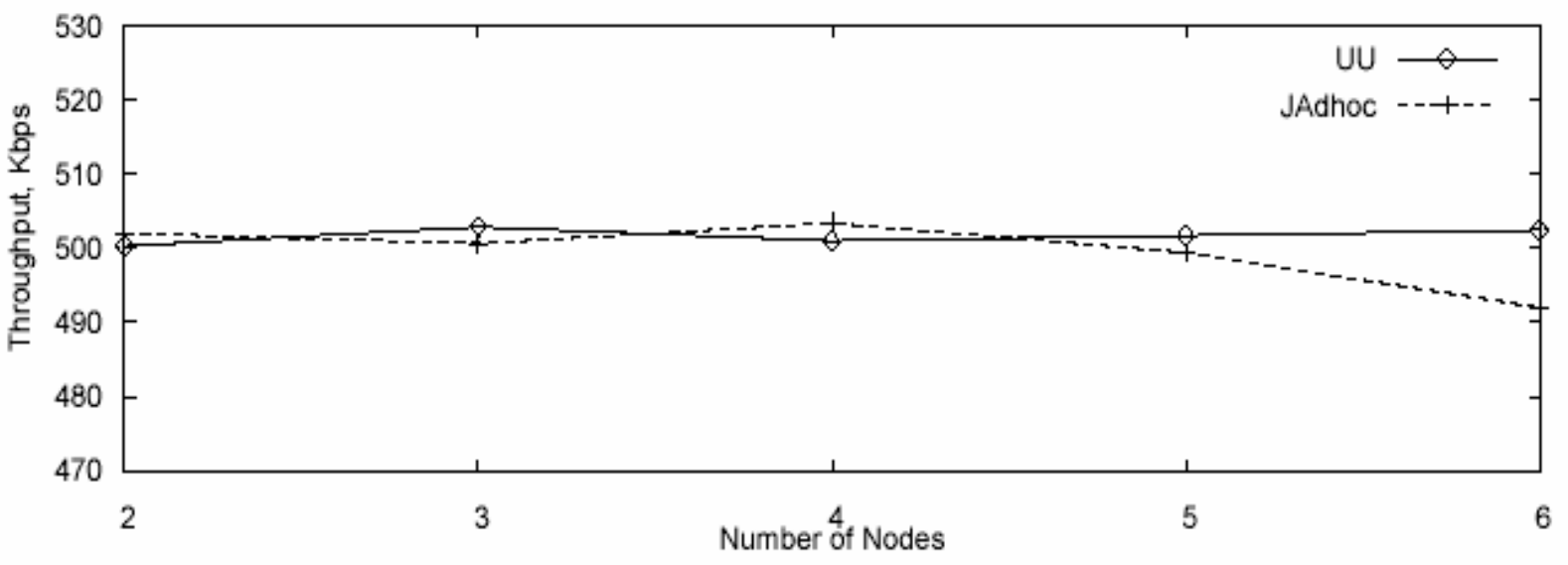
Throughput between 4 nodes



- UU & JAdhoc have higher throughput (than the given throughput) at the beginning, due to the release of buffered packets
- Causes packets to be lost due to the higher data rate at the link layer, while releasing the buffered packets

UDP Results Analysis

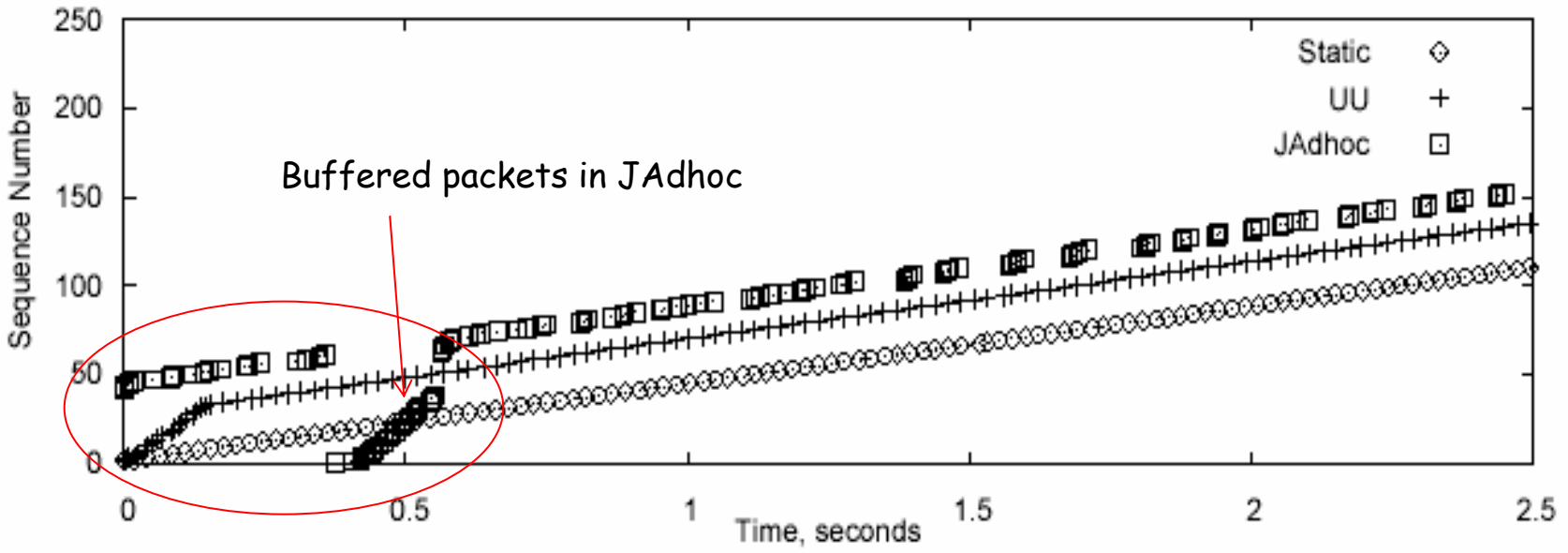
Actual load for an offered load of 0.5Mbps vs Number of nodes



- Average throughput of both UU & JAdhoc is ~ 0.5 Mbps
- JAdhoc has less throughput with a 6 node setup

UDP Results Analysis

Packet Sequence Number Vs Time



- Linear in Static mode
- UU releases buffered packets before allowing other packets (packets go just after setting routes)
- JAdhoc has out of order packets due to user space buffering (Other packets come first before the buffered packets)

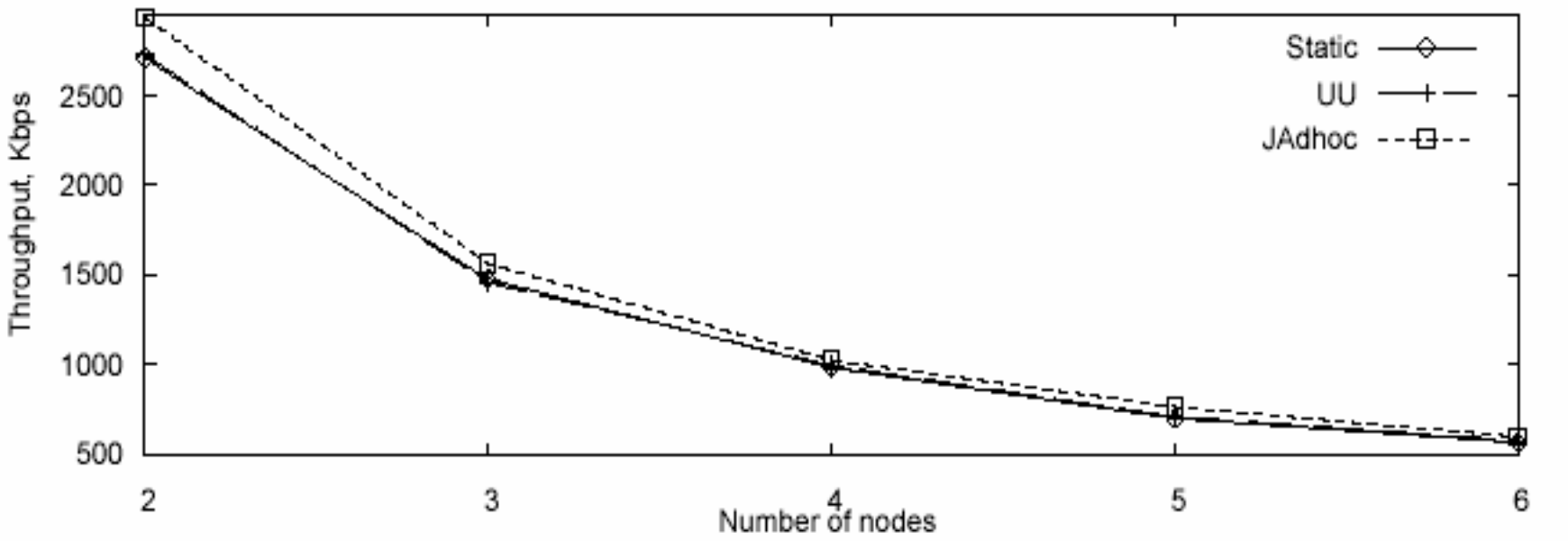
UDP Results Analysis

- UDP Performance in general
  - Static > UU > JAdhoc
  - JAdhoc Route discovery time is higher due to running at the user space
- Issues to be improved with JAdhoc
  - Buffering

TCP Results Analysis

TCP Results Analysis

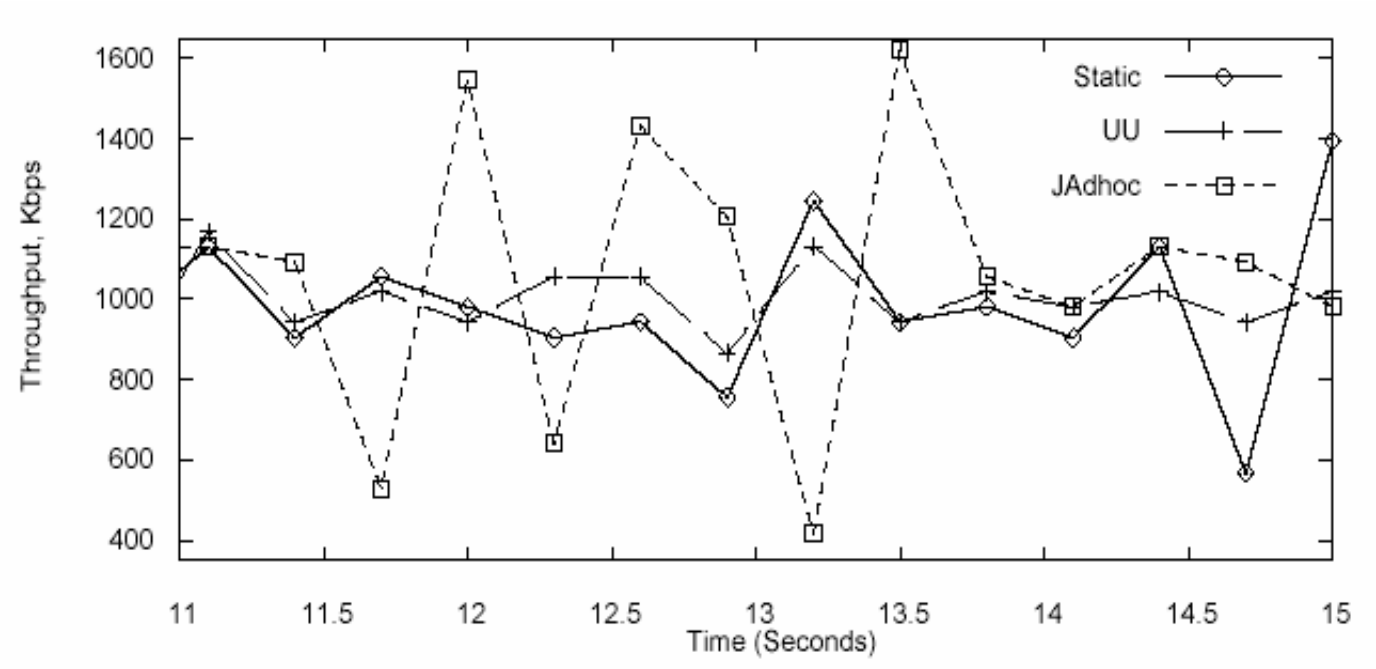
Throughput vs Number of nodes



- Less with the increase of nodes
- Contrast to UDP throughput, JAdhoc shows the highest throughput for TCP
- Closer analysis of TCP throughput of both UU and Static shows that UU has a higher throughput than Static

TCP Results Analysis

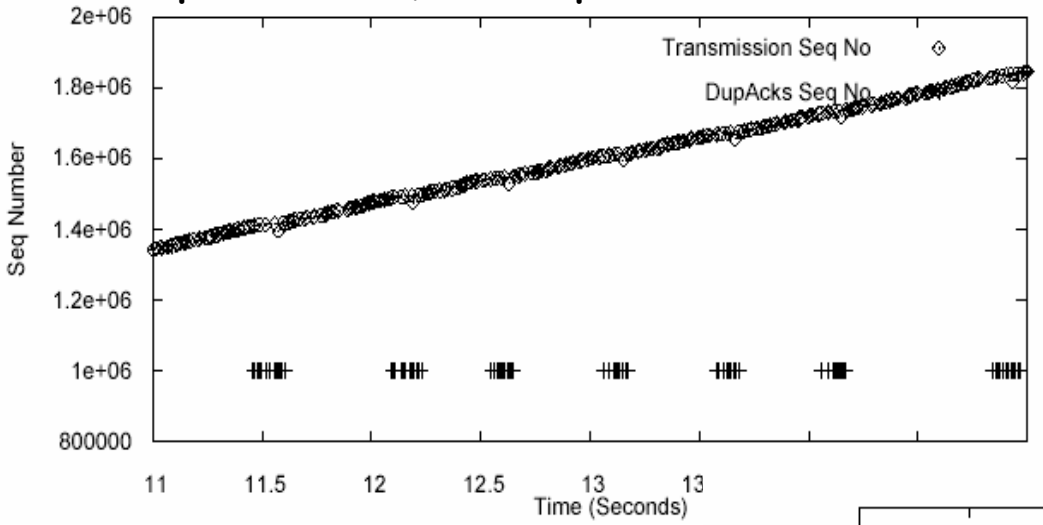
Throughput between four nodes



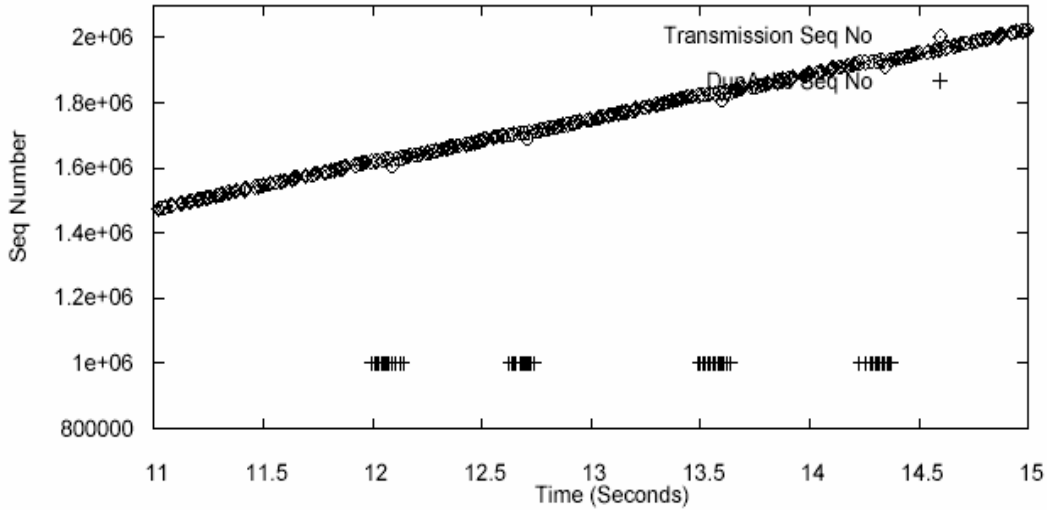
Reasons for the Throughput variations are investigated by analyzing TCP parameters dupacks, CWND & RTO for each mode of communication

TCP Results Analysis

TCP Flow packet trace, with dupacks for Static mode Packet Trace



JAdhoc – Packet Trace

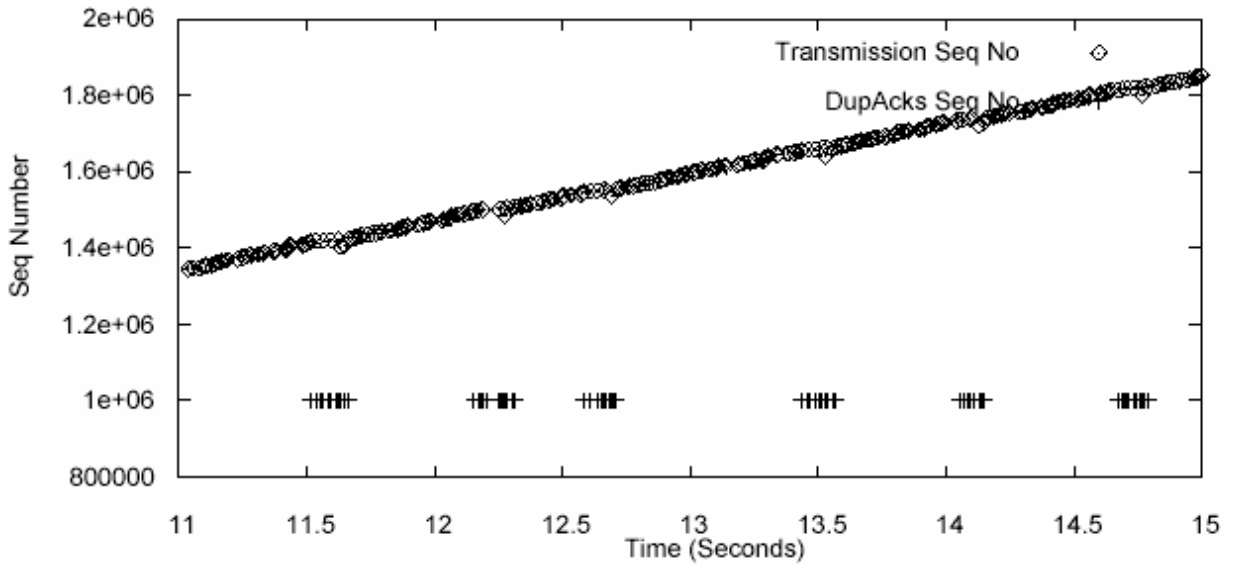


DUPACKS

- Higher in Static > JAdhoc

TCP Results Analysis

TCP Flow packet trace, with dupacks for UU mode Packet Trace



DUPACKS are higher in Static > UU > JAdhoc

TCP Results Analysis (CWND & RTO)

CWND (Slides 34 & 35)

- Variation of CWND

  Static > UU > JAdhoc

- Frequency of throughput variation is higher in Static

RTO (Slides 36 & 37)

-Variations of RTO

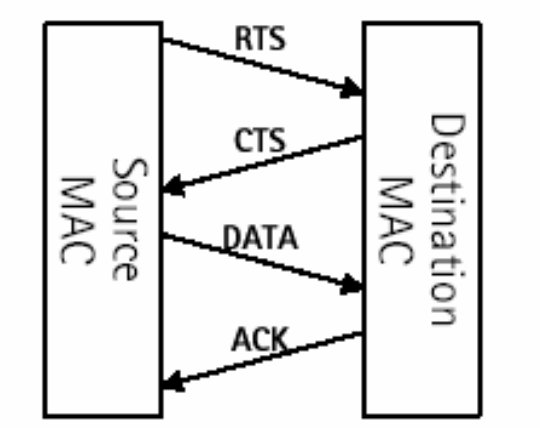
  JAdhoc > UU > Static

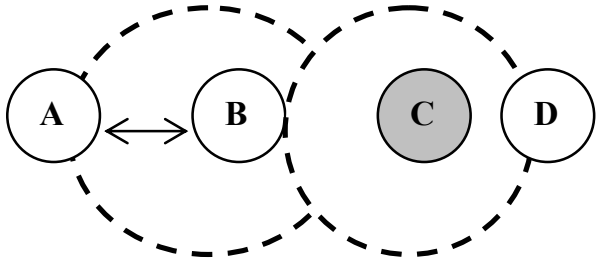
- In Static, more packets sent to link layer
- Therefore, higher packet loss at the link layer

TCP Results Analysis

- JAdhoc, being a user space program,
  RTO is higher for TCP communication
  TCP itself adjust with RTO variations and put less packets, causing less losses at the link layer

- TCP throughput (in general) is mostly degraded in ad hoc mode, due to the Exposed Node Problem





1. B sends data to A
2. B sends RTS to A, C also receives it
3. C sets NAV as carrier busy
4. C can't send data to D simultaneously
5. C is an exposed node during data transmission between A & B

Additional Observations

*1) Effect of "Hello Interval*Hello Loss"*

HelloInterval*HelloLoss is Lower, the probability of Route Breaks are higher at higher data rates

Solutions:

Better to use Link layer info to detect neighbor

Use Higher value for Hello Loss

in static n/w that uses higher data rates

2) Lifetime of the reverse route

MinimalLifetime = (current time + 2*NET_TRAVERSAL_TIME - 2*HopCount*NODE_TRAVERSAL_TIME)

Route Lifetime = 140ms

(HopCount=3,NODE_TRAVERSAL_TIME=10, Net Diameter =10)

Consider the processing delay for the MinimalLifetime

Further Investigations

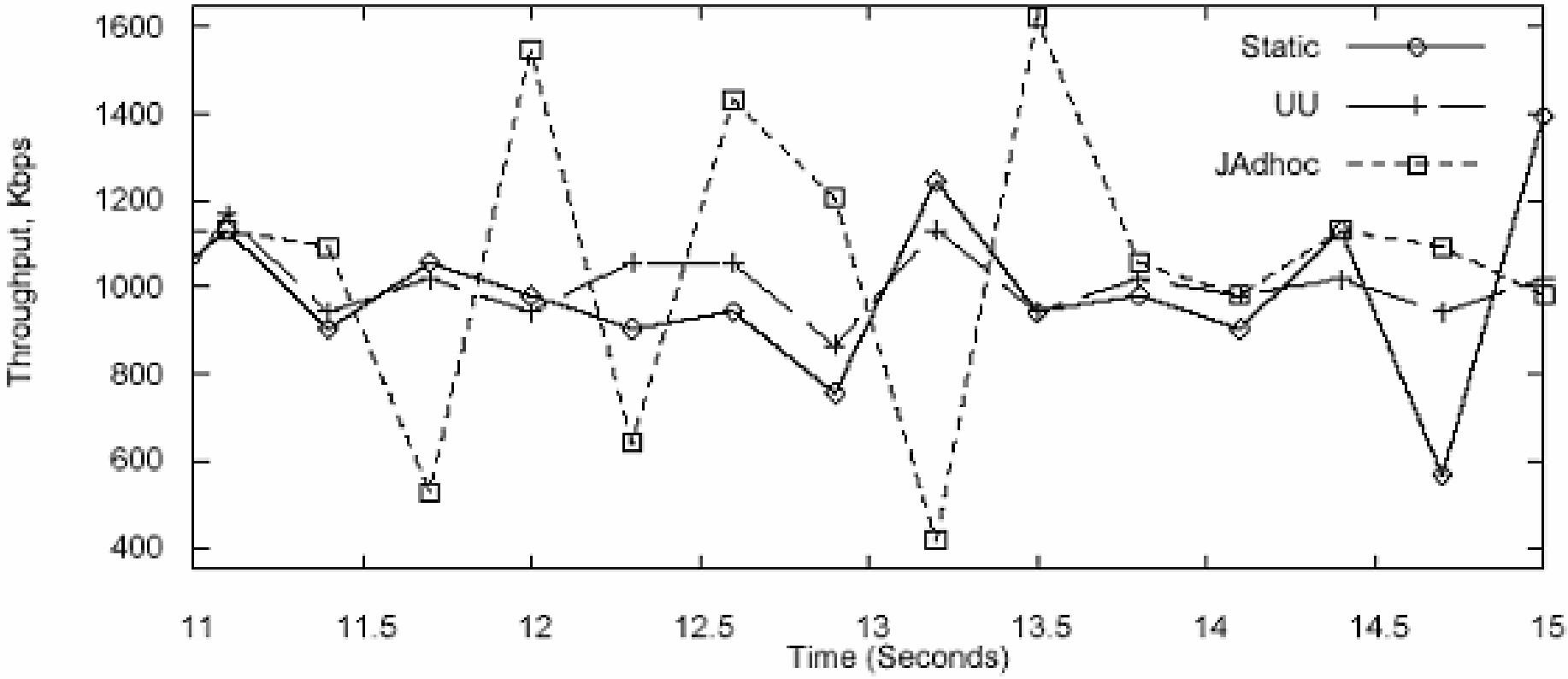
- Consider the mobility in the same set up
- Variations of results with the background traffic
- Effect of AODV parameters

Conclusions

- Architecture influences performance, due to following,
  - the way packet buffering is done in User Space implementations - results in initial delays and unordered packets
  - Packet Processing delay
- Transport layer performance can be improved by fine tuning AODV parameters to suit the topology (static or mobile)
- UU
  - Linux Platform
  - Performance is better due to use of Netfilter
- JAdhoc
  - Buffering has to be improved
  - Can easily be made to work across different platforms
  - Works on Linux/Windows XP & 2000
  - Possibility to be adopted to run on Java based mobile devices

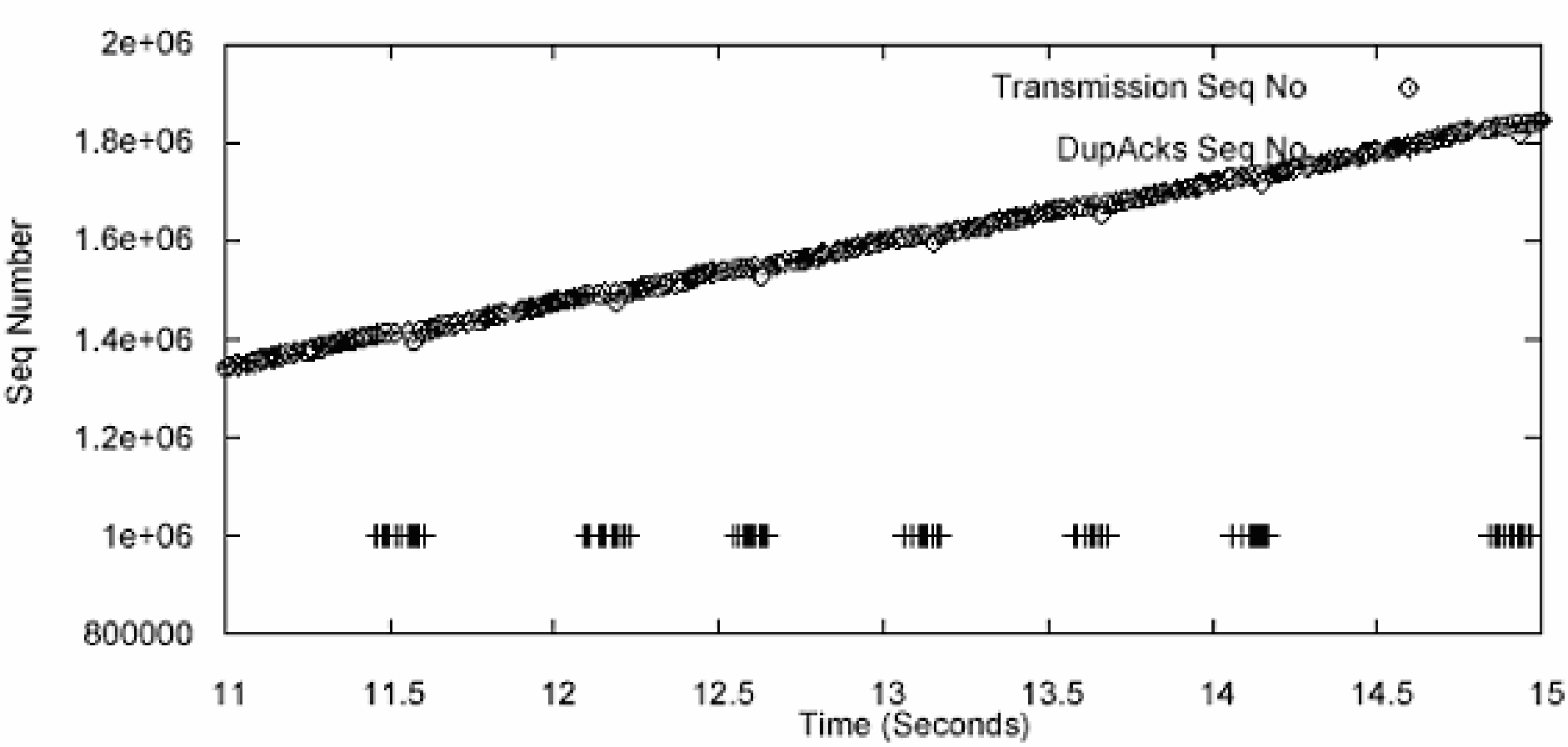
Rest of TCP Results

TCP Results



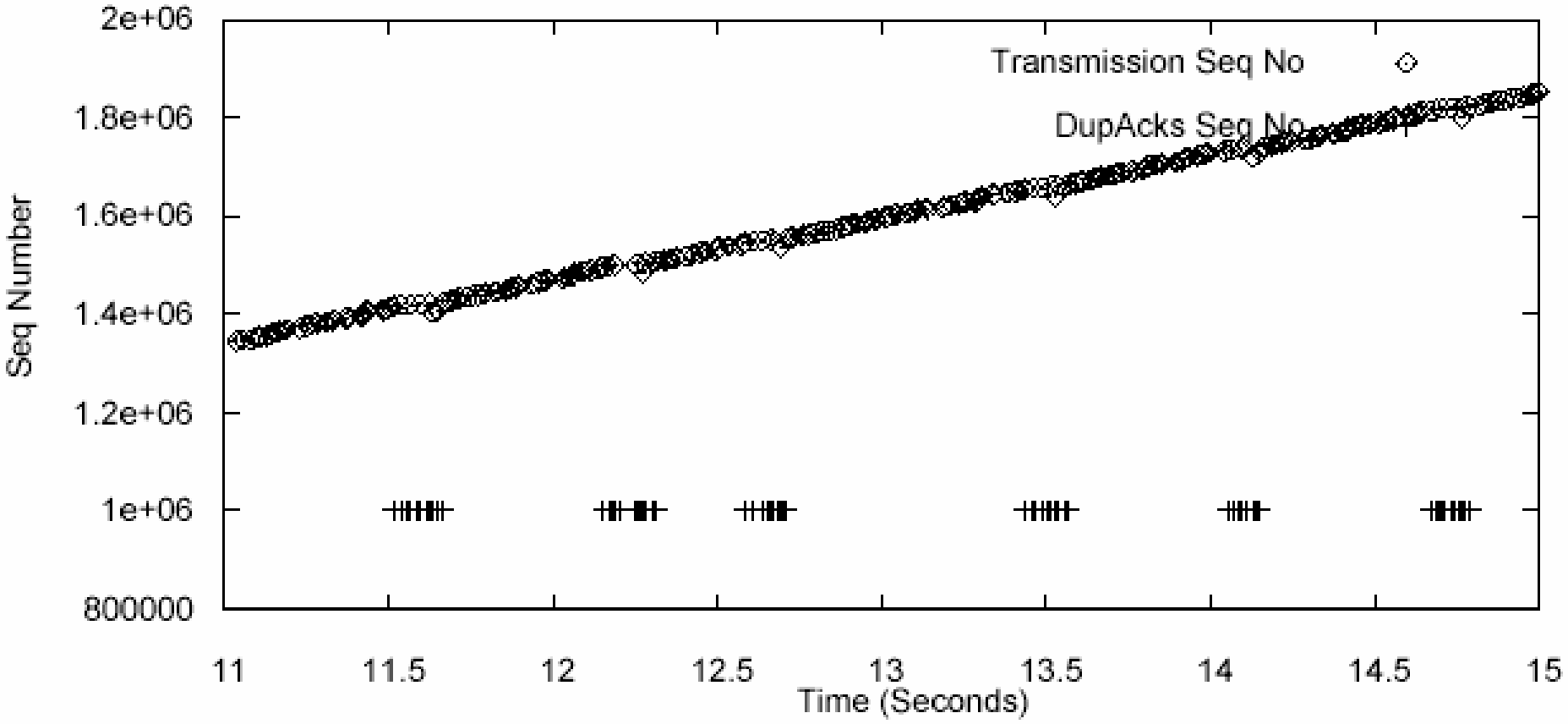
TCP throughput variations between 4 nodes

TCP Results



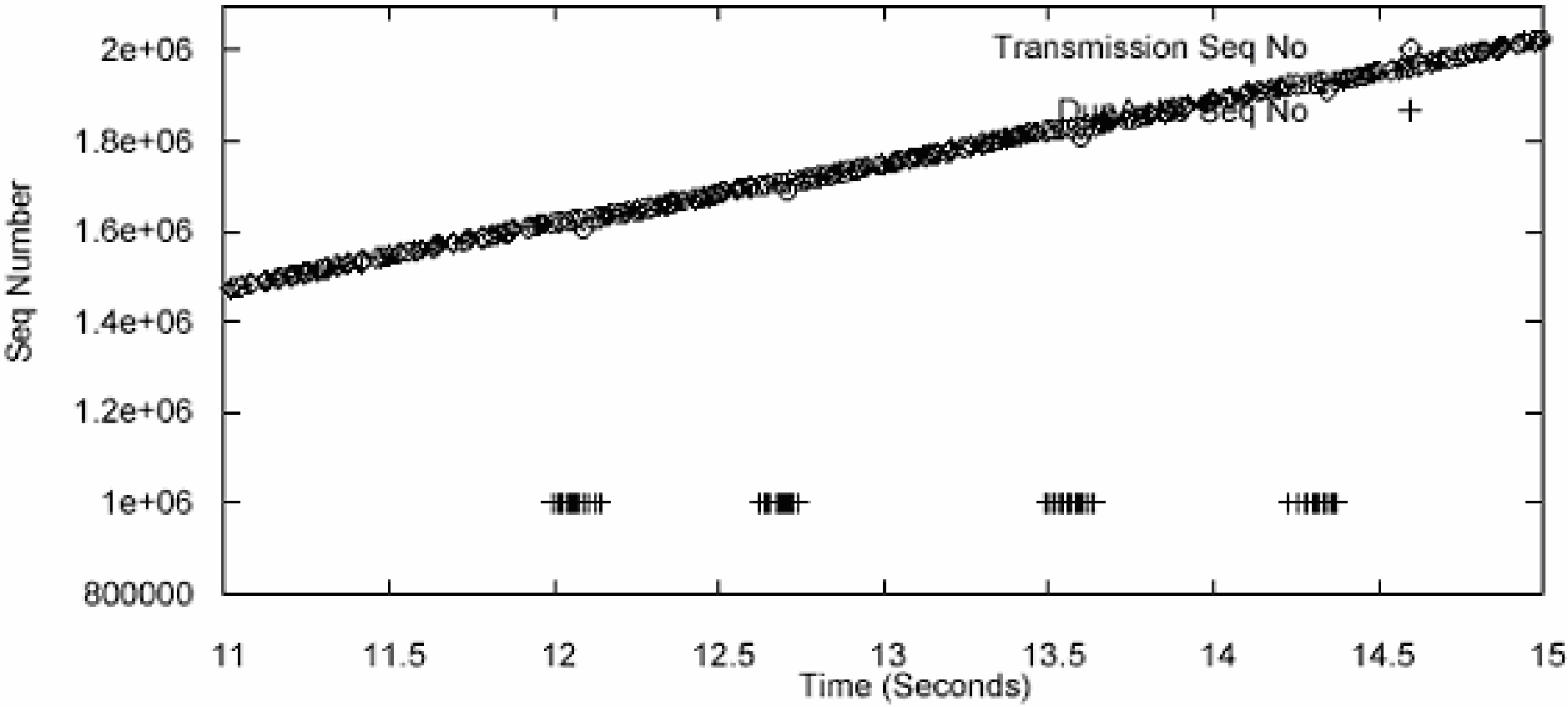
TCP Flow packet trace, with dupacks for Static mode

TCP Results



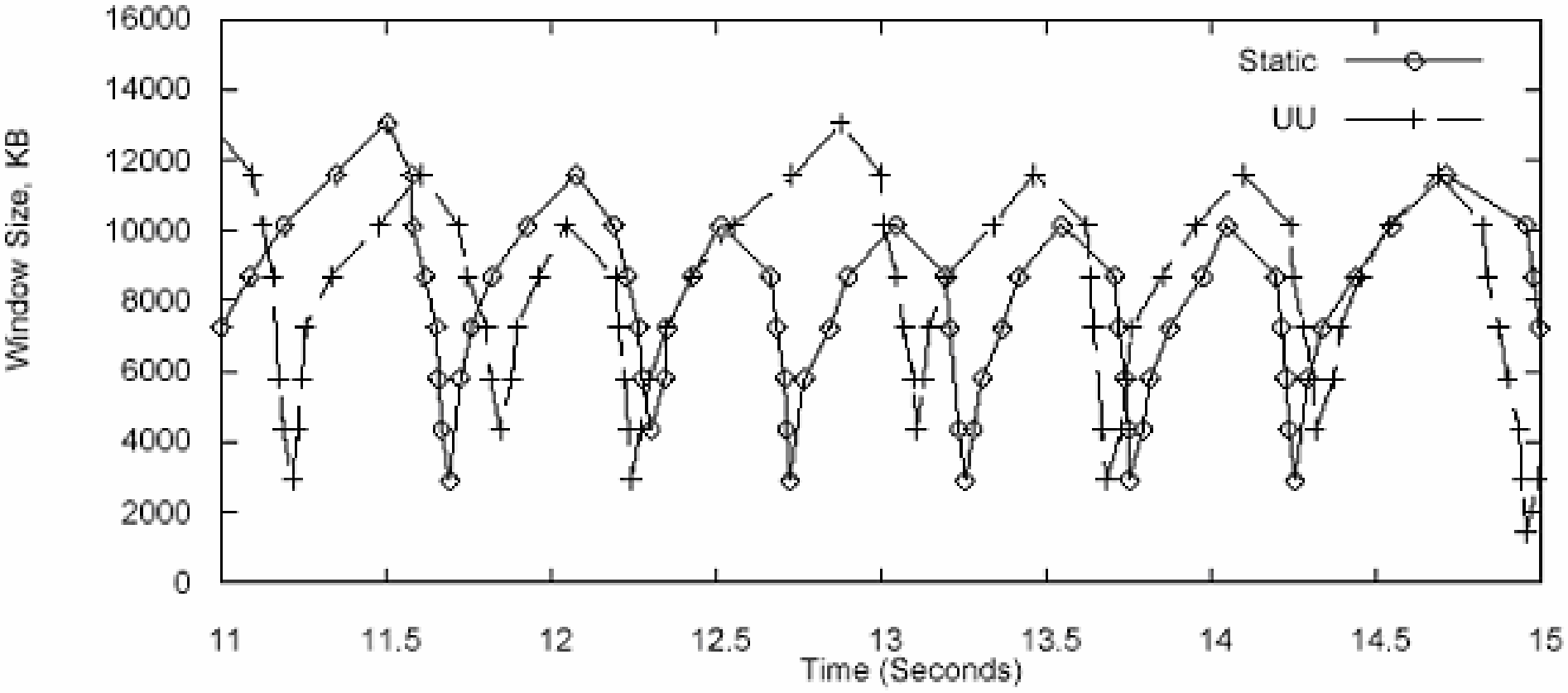
TCP Flow packet trace, with dupacks for UU mode

TCP Results



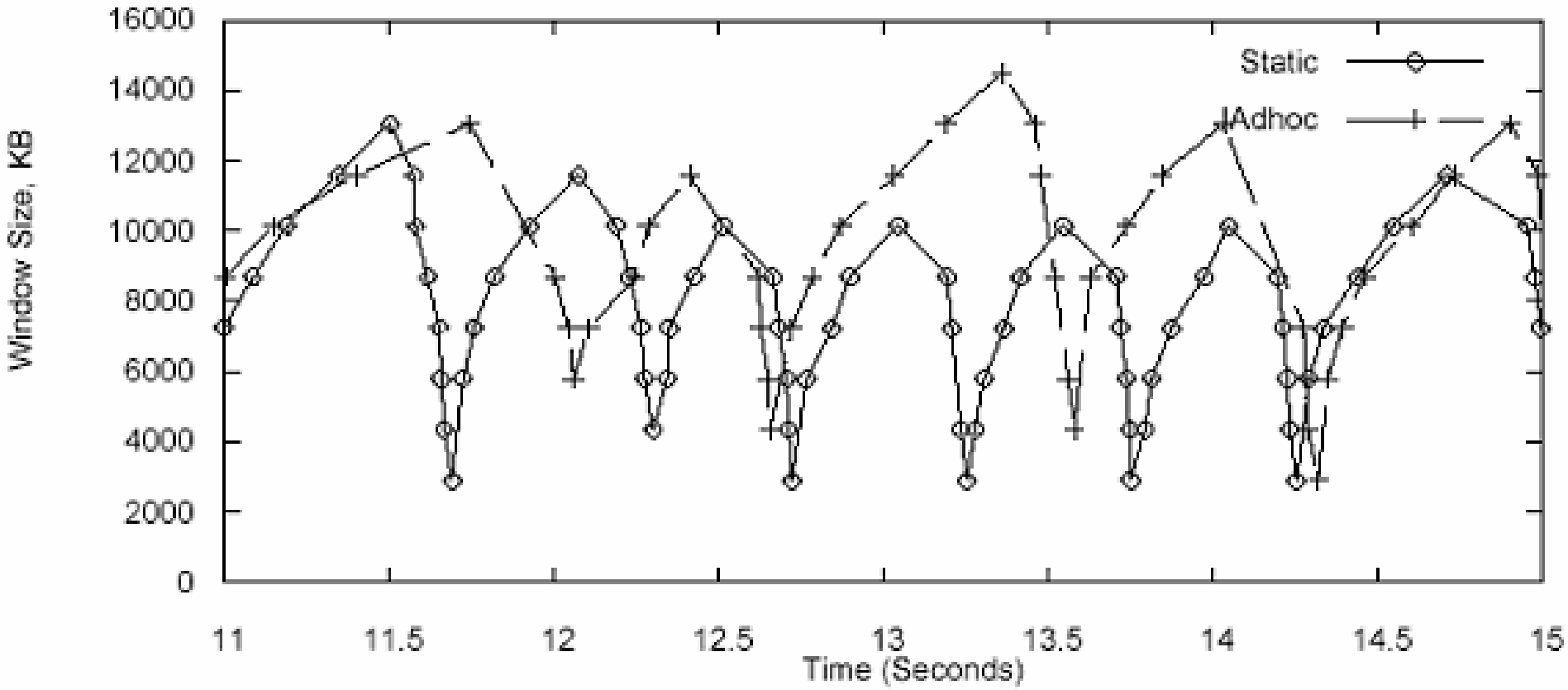
TCP Flow packet trace, with dupacks for JAdhoc mode

TCP Results



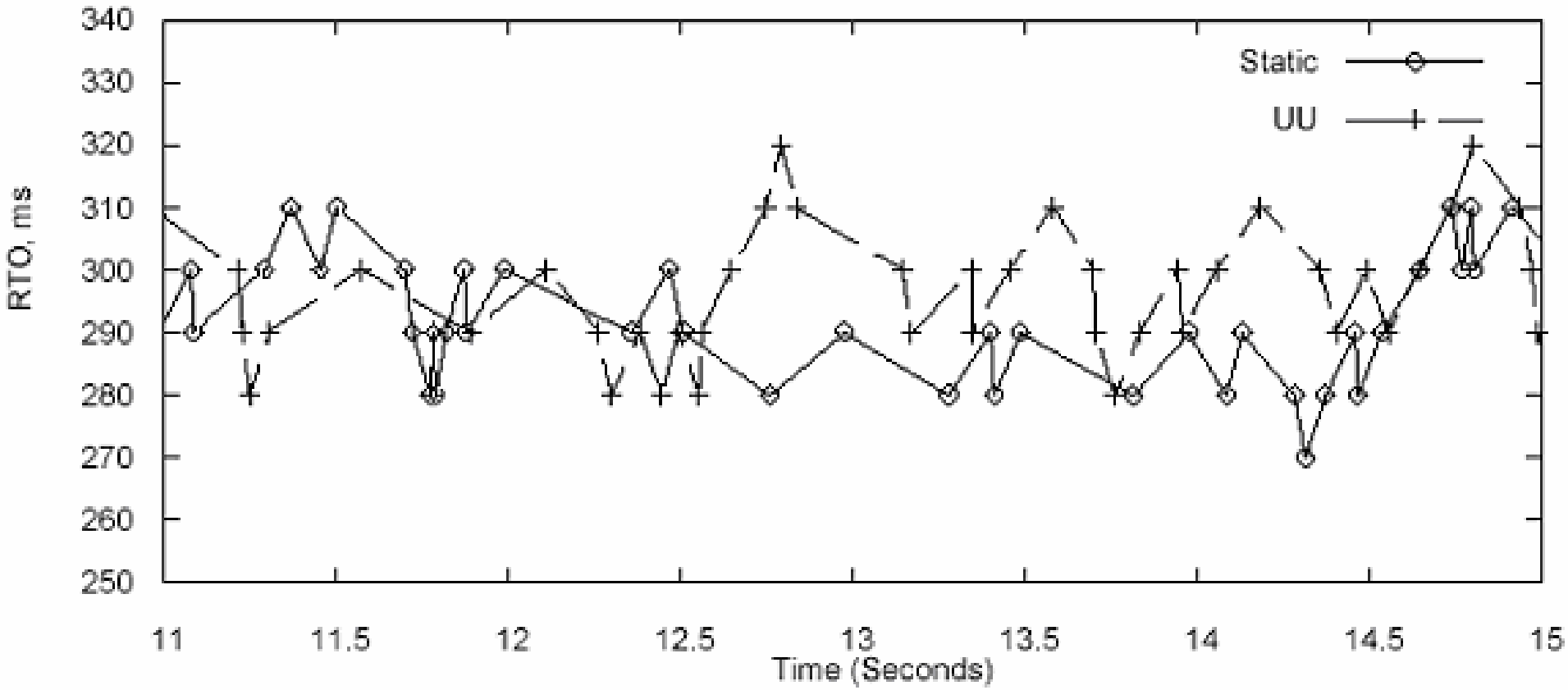
Variation of cwnd for Static & UU modes

TCP Results



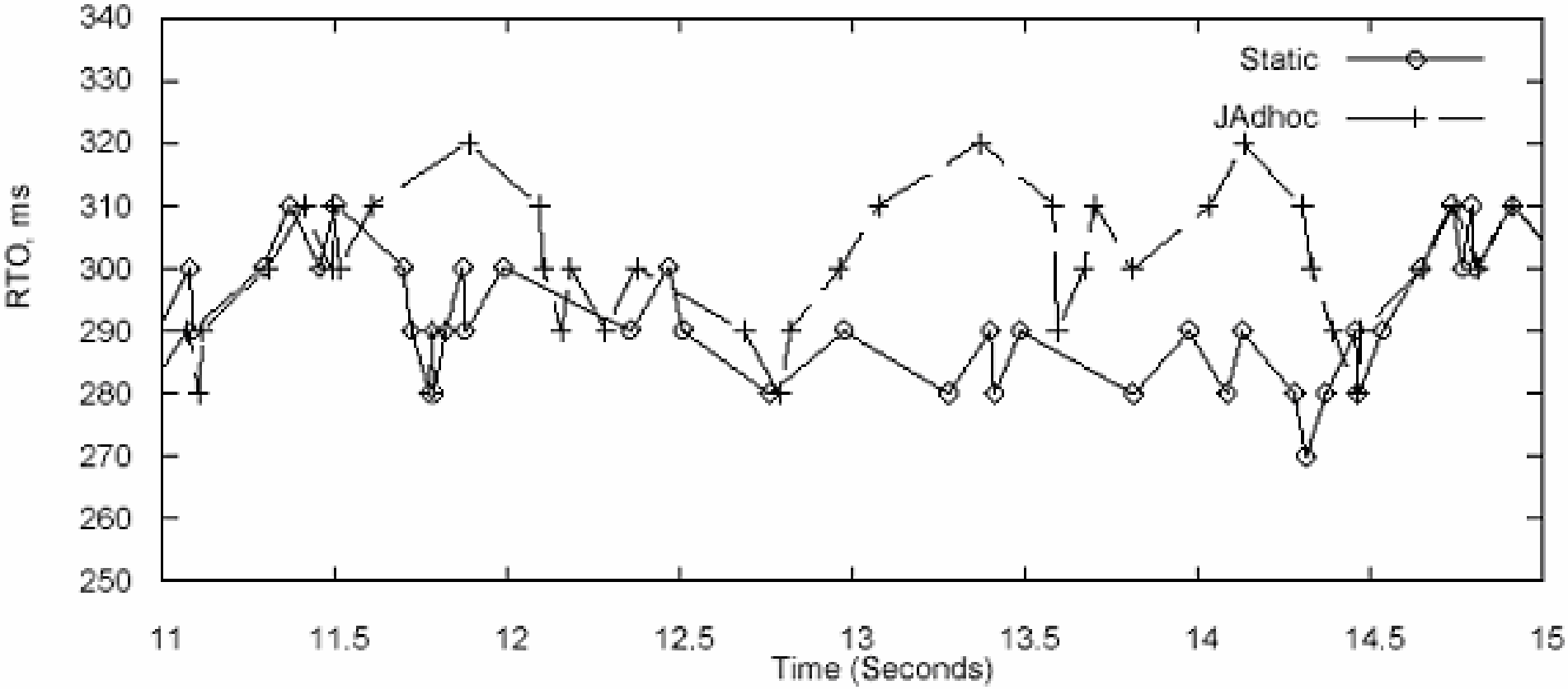
Variation of cwnd for Static & JAdhoc modes

TCP Results



RTO Variations for Static & UU modes

TCP Results



RTO Variations for Static & JAdhoc modes