Methodologies for Measuring IEEE 802.11 Networks with Application

Ng Yat Fai

Master of Science in Information Technology

The Hong Kong Polytechnic University

2008

Statement of Authorship

Except where reference is made in the text of this dissertation, this dissertation contains no material published elsewhere or extracted in whole or in part from a dissertation presented by me for another degree or diploma.

No other person's work has been used without due acknowledgement in the main text of the dissertation.

This dissertation has not been submitted for the award of any other degree or diploma in any other tertiary institution.

Name: Ng Yat Fai

Dated: 01/09/2008

Acknowledgements

I would like to thank my supervisor – Dr. Rocky K.C. Chang, for his guidance, encouragement, advice and directive in the preparation of this report. With his kindly help, my work has been completed as expected. This is one of my invaluable experiences being his student and willing to be again at the further study.

Also, I would like to thank the members of research team, Dr. Daniel Luo, Mr. Edmond Chan, Miss Grace Xie and Miss Samantha Lo, for help and support. With their valuable idea and comments, most problems in progression have be solved and new consideration are added to sharpen the result.

Finally, I would like to thank my family for their support and forgiveness for neglecting them. They let me concentrated in my study.

Thanks again for all support and encouragement.

Abstract

This study is devoted on the effectiveness of various methodologies for measuring IEEE 802.11 network. Besides of the methodology explanation, I have developed a set of new tools for measurement control and data presentation. Also, the new "Auto-Selecting the Rate Adaptation Algorithm" (ASRAA) is designed for automatic / intentional Rate Control Algorithms (RCA) selection in wireless network driver. And, a RCA competition is accomplished for network performance analysis in heavy traffic environment.

Most likely, successful network measurement is relied on the effective data capturing method. These captured information are the original source for any decided analysis. Currently, there are two types of capturing arrangement. One is active capturing system which is running in the member node in the measured network. Thus, the capturing overhead might affect precise measurement in that node. The most disadvantage is that it is appliable for domestic network only. The other is passive sniffing system which is running in the additional invisible node for the target network, i.e. no link state is registered in the network. There is no overhead to be added to any former nodes in the original network. It can be

applied on any touchable or untouchable network. Because of this, the passive sniffing is focused in this study. The Multiband Atheros Driver for WiFi (MadWiFi) [4] for Linux and the AirPcap [1] for Windows are common passive sniffer. The comparison shows that the MadWiFi is capable to capture more data in a congested network. But, the AirPcap is very good for exploring strange network.

The "RemoteCall" and "RemoteExecute" are new tools to control large testbed, i.e. network with many nodes. They are worked as client / server structure. Control signal are transmitted via wired IP network that is separated from the measured wireless network. No overhead is added to influence the target network. They are small JAVA program to be run on all common platform. The "RemoteCall" is the centralized controller for test procedure setup and execution trigger. The "RemoteExecute" is actual command setup and execution master on every nodes in the network.

Meanwhile, the "XGraph" is a Java based IEEE 802.11 network traffic analyzer, it reads in libpcap formatted output file and generates time-aligned graphs for traffic analysis. Multiple nodes are supported. The graph includes rich information, such as data / ACK packets count, transmission rate, packet loss detect, signal strength and etc.

Also, the "Auto-Selecting the Rate Adaptation Algorithm" (ASRAA) will be introduced as a new concept of rate control. Usually, there is a single RCA to be

implemented for each network driver. The ASRAA is a combination of multiple RCA logic and provides a policy based selection mechanism to choose a RCA for each transmission. The Auto-RCA-Select engine is counting on statistical status and current application behavior, such as size of data block and TCP / UDP port for application classification. The on-line packet analysis is deployed to predict the outgoing packet pattern which is useful for an appropriate RCA selection for effective transmission.

Finally, I use all of above to measure the performance of various Rate Control Algorithms (RCA) in a busy network. It is different from usual performance analysis. They check the RCA against distant or signal strength for peer-to-peer network. I will check for channel competition on multiple nodes network. It is because the single client network is not a practical operational environment. In my study, there are four RCAs to be selected for comparison. They are ARF [7], AMRR [8], ONOE [3], and Sample [6]. ARF is the most referential RCA and is implemented in my testbed. AMRR, ONOE and Sample are built-in RCA for MadWiFi driver. The objective of the measurement is to illustrate their behavior in a congested network under different scenarios, such as all nodes using the same RCA or different RCA for each node. The result shows that the network performance is seriously degraded when four nodes make heavy transmission to wireless access point. It is far from usual expectation for busy wireless network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is focused on effective methodology for measuring IEEE 802.11 wireless network, Rate Control Algorithms (RCA) efficiency against network population, and the "Auto-Selecting the Rate Adaptation Algorithm" (ASRAA) design.

The proposed methodology includes testbed setup, sniffing device selection, control tool, and data presentation. The control tool are designed to be worked on large testbed. It provides scheduling, remote command setup and execution triggering.

The experiment on RCA efficiency against network population is done on an extreme busy network. The objective is to observe overall performance and fairness between ARF [7], AMRR [8], ONOE [3] and Sample [6].

The "Auto-Selecting the Rate Adaptation Algorithm" (ASRAA) is new approach on RCA design. The heart of ASRAA is the Auto-RCA-Select engine which will analysis the outbound packet to classify the current application type plus statistical status then makes the RCA selection.

1.1 IEEE 802.11 and RCA

IEEE 802.11 wireless network is designed to run on open media, the air. There are many influence which may come from anywhere. Example, IEEE 802.11b and 802.11g are the most popular wireless network standard. They are using the same 2.4GHz band but with different maximum throughput, 11Mbps and 54Mbps. This frequency band is also used for most wireless phone and Bluetooth device. It is also covered by many microwave appliance. Encountering of influence will degrade the network performance and it is good idea to record those noise in the measurement.

Rate Control Algorithm (RCA) is an important component in 802.11 network driver. It controls the transmission rate for all outgoing traffic. The RCA affects the overall network throughput directly; faster rate yields higher throughput under normal conditions. There are three major functions to be provided by RCA.

1. Initialization — When an NIC is brought online or an existing Access Point (AP) association changes, all the rate control parameters must be initialized.
2. Request Rate Set — The RCA logic will check on the current and historical states to decide which rate to use for the outbound transmission.
3. Status Update — After the packet is sent to lower layer handler, the status should be returned whatever the transmission is successful or not. The RCA logic will record and update the status for further operation.

There are various RCA implementations, they can be classified as following ways:

1. Fixed — Only a single transmission rate is used; no dynamical change is allowed.
2. Aggressive — It is designed to be sensitive to the state change, and the threshold for the rate change is fixed. Example: ARF [7].

3. Adaptive — The threshold for the rate change can be adapted according to the current states. Example: AMRR [8].

4. Statistical — The rate change is performed based on credit calculation or counting on successful and failed transmissions. Example: ONOE [3] and Sample [6].

Five RCA implementations in this thesis shown as follows:

1. ARF [7] — Auto Rate Fallback is the earliest RSA. Many other RCAs are based on this logic, such as Adaptive ARF and AMRR. It is an aggressive algorithm to try use higher rate and sensitive for transmission failure. Once the number of consecutive failure reach the preset threshold, the next lower rate will be used. The sinking threshold is set to two in our implementation. The rate will raise when ten consecutive good transmission which means no packet retransmission to be performed. The rate raising threshold is fixed and usually be small number. The rate will also be raised up after certain period of idle state. Because of its aggressive behavior then it may have performance gain on some situation.

2. AMRR [8] — Adaptive Multi Rate Retry is one of standard RCA in MadWiFi. It is the refinement of ARF. The rate change logic is similar to ARF. The major different is the rate raising threshold is exponential increased for consecutive fail transmission. It slows down the rate raising when the network is highly unstable, i.e. there are many fail transmission.

3. Fixed — The full manual control packet transmission rate. The rate is set to fix until user change is taking place.

4. ONOE [3] — It is credit based algorithm. The credit value is determined by the number of successful or fail transmission. The value will be raised for each successful transmission and reduced for each fail transmission. When the credit value is reached the raising threshold, the next higher rate in the list will be used. Similarly, next lower rate will be used for the credit value below the lower threshold. Otherwise, the rate is no change.

This lets the logic is insensitive for instant environment change, low jitters. It may be good for time sensitive application, such as VoIP.

5. Sample [6] — It is the most complex RCA in MadWiFi. The rate selection is based on the statistical matrix. It is formed by three frame's size groups and those supported rates. Each cell contains counters for good (ACK received), fail (time out), retry (retransmission). Before transmission, the rate with highest score in same frame size group will be used. And, the status counter will be updated once the transmission to be completed or failed. It seems the smart algorithm with some kind of adaptation between appropriate transmission rate and the frame's size.

1.2 ASRAA — Auto-Selecting the Rate Adaptation Algorithm

The Auto-Selecting the Rate Adaptation Algorithm (ASRAA) is an enhanced RCA for 802.11 network. I have implemented it into the Multiband Atheros Driver for WiFi [4] (MadWiFi, version 0.9.2) for evaluation. The core of ASRAA is the Auto-RCA-Select which is on-line packet analyzer and RCA selection engine. It provides the mechanism to select RCA during run-time. The RCA selection can be initiated manually or automatically. Manual selection is via the /proc standard file system in Linux, which is the root of hierarchical register for system and device information. It is useful for test cases that users can take a full control on RCA deployment for different scenarios. The automatic selection is an idea of the smart function to apply appropriate RCA for specific application or situation. That function will analyze the packet detail and recent environmental status for decision making. All packet's header fields (type, protocol, source and destination, etc.) and available statistic (average RSSI [5], transmission power, various counters, etc.) can be counted into selection processing. It is flexible for future extension.

The ASRAA includes AMRR, ARF, Fixed, ONOE and Sample. The first two are popular algorithms which are used in many 802.11 implementations. The fixed rate is used for a full manual control on the transmission rate. The last two are maintained by MadWiFi initially, and they are not common in other packages.

All of five RCA have their own character and may have advantage for various situation. The simple classification of them are — "ARF" is aggressive on raising rate, always trying a higher rate. "AMRR" is similar to ARF but less aggressive. "Fixed" is used for reference or as RCA off. "ONOE" is conservative, not sensitive to the environmental change. "Sample", on the other hand, is an adaptive algorithm based on packet size.

1.2.1 Auto-RCA-Select

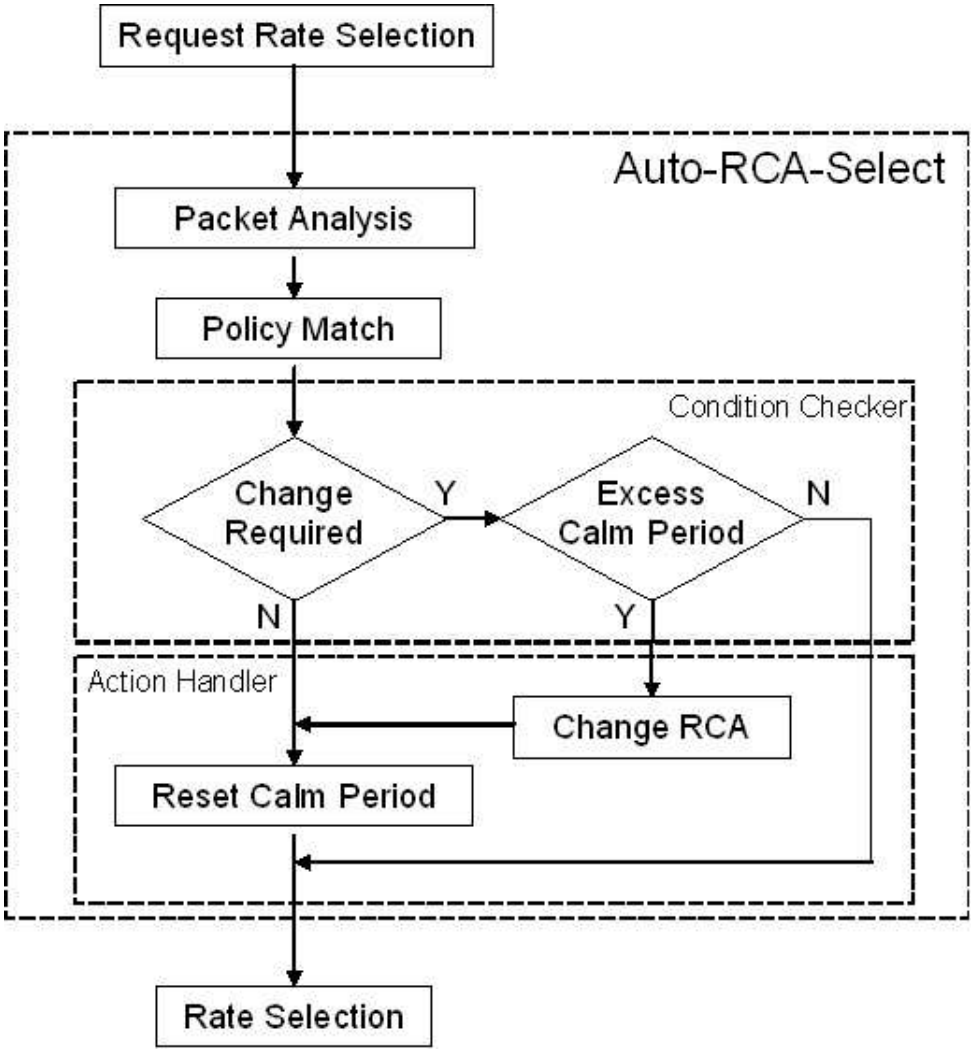


Figure 1.1: Auto-RCA-Select for ASRAA

The block diagram of the "Auto-RCA-Select" module is shown in the Figure 1.1. Its major components are:

1. Packet Analyzer — Analysis the packet from upper layer. Detect the packet type and its properties, such as IP address, TCP or UDP port number, etc.
2. Policy Matcher — Use above information to match the policy table.
3. Condition Checker — Check the result from above step. If change is required, check again the expiration of the Calm Period.

4. Action Handler — According to the Condition Checker direction.

And, the key parameters are:

1. Policy — Predefined refer table for RCA select.
2. Clam Period — Control the RCA switching frequency. Selected RCA will be used for decided period and will be switched to another RCA when there is no more matched type of transmission occurred in that period.

1.2.2 Design Concept

In original wireless network, the single fixed rate is used for all kind of transmission. There is no idea of rate control function. After that, multiple rate networks are built for high speed transmission, such as 802.11 transmits packet in 1 or 2Mbps, 802.11b works from 1 to 11Mbps, 802.11a/g runs up to 54Mbps, and recent 802.11n is designed for over hundred Mbps. All of them need a function to select the transmission rate. Then, Rate Control Algorithms (RCA) becomes the standard function on driver implementation for transmission rate selection.

For various rate selection strategy, the obvious aim is to elect appropriate rate for transmission that is adapted to the environment. Usually, try to keep in higher data rate for performance. Or use lower rate to secure the reliability of transmission in high .

There are three types of strategy to be used for rate adaption.

1. Aggressive — Based on simple counting of success and failure on consecutive transmission. Raises the rate after certain numbers of successful and reduces the rate when error occurred. The check counter is usually small. It is aggressively to keep trying on high rate transmission. The example algorithms are “ARF” and “AMRR”.

2. Dispassionate — Slower reaction on success and failure, the threshold of rate change is high. It needs longer time to confirm the situation. The objective is to smooth the delivery interval. The example algorithm is “ONOE”.

3. Classification — Packet type is concerned in rate adaptation. As example, “Sample” will consider the packet size to be transmission. There are three groups of statistical information for packet size in range of 250, 1600, 3000 bytes. Rate selection is independent in these groups.

In the ASRAA, the idea of selectable RCA is another new strategy of adaptation. The RCA selection is based on sending packet analysis, such as size, type, destination, and etc. The most new idea is the consideration of sending application to be included in lower level rate selection algorithm. It is possible to predict transmission pattern by understanding of the application behavior. For example, FTP will send series of large packet. VoIP will send small packet in constant interval. It is deserved to make further study to illustrate the application depended rate adaptation algorithm.

1.3 Type of Sniffing for Wireless Network

In general, there are two types of capturing arrangement. One is active capturing system which is running in the member node in the measured network. Thus, the capturing overhead might affect precise measurement in that node. The most disadvantage is that it is appliable for domestic network only. The other is passive sniffing system which is running in the additional invisible node for the target network, i.e. no link state is registered in the network. There is no overhead to be added to any former nodes in the original network. It can be applied on any touchable or untouchable network. Because of this, only the passive sniffing is discussed in this study.

There are four types of passive sniffing setup. The summary is in Page 11 Table 1.1.

1. Virtual Sniffer (Figure 1.2) — It is software based virtual sniffing device. Packets will be copied from I/O stack to monitor buffer locally. For example, MadWiFi has capability to duplicate traffic from layer-two I/O queue to its Monitor VAP. User can use general application, such as tcpdump or Wireshark, to capture the traffic from the Monitor VAP while the Station VAP is still working. Both VAP are linked to same physical device.

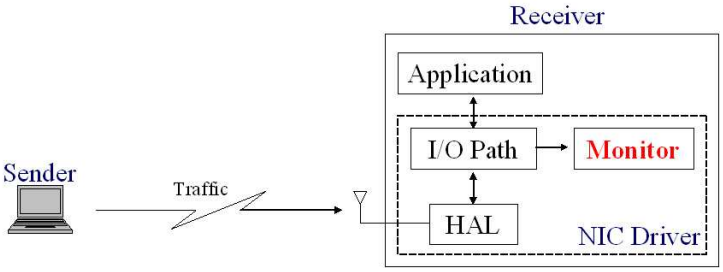


Figure 1.2: Virtual Sniffer

2. Internal Sniffer (Figure 1.3) — The dedicated sniffing device is installed with a normal NIC in same machine which is one of measured node. They are sharing all system resource. Network traffic is captured from the media.

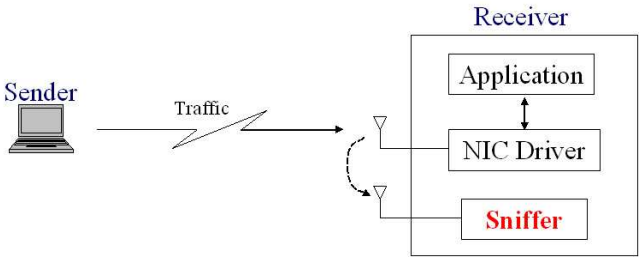


Figure 1.3: Internal Sniffer

3. External Sniffer (Figure 1.4) — The sniffer is installed into a dedicated machine which is not a part of measured nodes. No packet will be generated from this machine.

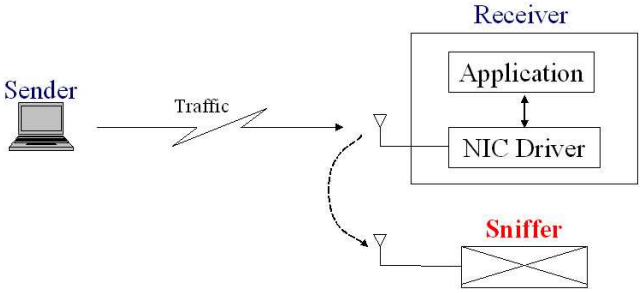


Figure 1.4: External Sniffer

4. Multiple Sniffers (Figure 1.5) — Co-operated multiple sniffing devices are used in the measurement. They may be installed in single or multiple machines. All captured data are expected to be merged together for analysis. This setup is most useful when the capability of the individual device cannot be guaranteed. But, time alignment between sniffers is difficult.

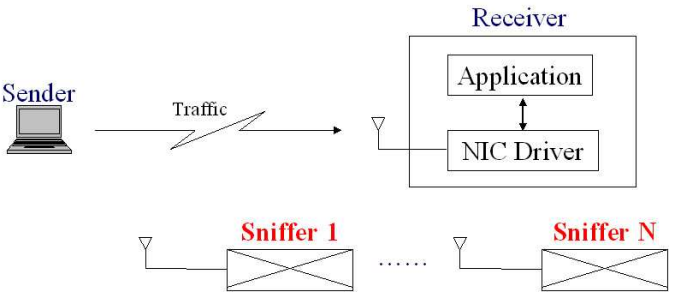


Figure 1.5: Multiple Sniffer

Type	Pos	Cons
Virtual	Easy to setup, no additional software will be used.	Local loop in Layer 2, not all packet types will be captured.
Internal	Sniffer is worked independently, external capture for all packet types.	Shared system resource, it is an issue for lower performance machine.
External	No hardware performance issue, may use different platform.	Extra station is requested to host the sniffing device.
Multiple	Sniffer capability issue may be overcome.	Extra hardware and software are requested, tool for merging and filtering data from multiple sniffers. Time alignment between sniffers is difficult.

Table 1.1: Type of Sniffing Setup

1.4 Wireless Network Measurement

Successful measurement is rely on careful planning. The plan includes scope of network, environmental consideration, testing tools, test cases, control tools, data collection method, analysis tools and presentation tools. All of above are well known steps and general idea for network measurement.

After my study on wireless network measurement techniques. The following highlights will help on precise measurement on complex network. Complex network means that there are many active nodes with different setup in a uncontrolled environment. Many unknown factors are affected the measured data. We should take case of these interferences to get evident analysis.

1.4.1 Correlative Factors

In general, measurement of single factor is fair enough for simple network test. For example, measure throughput between client and AP, signal strength in single node. But, it is not good enough for thoughtful analysis. The consideration of multiple factors and their correlation is necessary. Such as following factors are correlated and they will directly or indirectly affect the channel throughput in performance analysis:

- Transmission rate is the major factor in throughput analysis, higher rate means more data can be transmitted in same period of time in case of no disruption is there.
- Transmission rate is control by Rate Control Algorithm (RCA) which is usually build into wireless device driver and programmed to select an appropriate rate which is based on analyzed result from historical statistic. Example, counting of lost 802.11 ACK and retransmission are common reference for RCA logic.
- Lost ACK might be caused by interference or collision applied on the original data frame or the ACK frame. This affects the rate selection indirectly and trigger retransmission

that reduces the goodput.

- Interference effect is more serious when transmission signal is relatively weak or there are much higher environmental noise.
- Probability of collision is high in hidden nodes effect.

1.4.2 Testbed Control and Scheduling

The planning of control and scheduling is important for testbed running. It is especially for multi-nodes environment. No measurement can be finished without any control and scheduling. The basic definition of them are:

- Control — Describe the configuration for all components. Prepare the testbed initialization. Setup the task list for each nodes. Catch and handle exception case. Define the status logging mechanism.
- Scheduling — Define the timeline of the measurement, such as running sequence, start simultaneously or delayed start. Synchronize progression of all nodes.

1.5 New Tools - RemoteCall and XGraph

1.5.1 RemoteCall - Remote Control and Scheduler

This tool consists of two Java programs, the "RemoteCall" and "RemoteCallExecute". Combined with TELNET / SSH, full remote control and monitor are applicable for the testbed with large number of nodes. Centralized control will make the task configuration easier, no physical touch on each node for test setup. It makes test change faster. Status inspection can be done remotely, no need to survey node by node. Moreover, this remote control system is designed to work on any platform with Java Runtime support. It is not developed for this study only.

RemoteCall

Figure 1.6: RemoteCall user interface

The "RemoteCall" is the main program for remote control, tasks scheduling and test execution trigger. All control messages are sent through IP network. In all of my test, the control media is 100BaseT wired network. It is separated from the object wireless media, this eliminates the interference which is caused by remote control signal. Remote control commands are sent via TCP unicast for single host setup. The trigger message is sent via UDP broadcast to signal all clients to run the test. UDP is a lower latency protocol, no connection overhead as TCP.

It is the effective signaling for simple local area network, i.e. no routing is needed for peer communication.

There are two execution modes, normal and random delay. The normal mode, all nodes will start the execution simultaneously. The random delay mode, individual node will delay to start execution. The delay is randomized and within a specified range. The user interface is as Figure 1.6 in Page 14.

RemoteCallExecute

The "RemoteCallExecute" will be run on client sides. It collects test commands from the controller and executes them when it is triggered. The execution status can be displayed on screen or redirected to any standard output device.

1.5.2 XGraph - 802.11 Traffic Analyzer

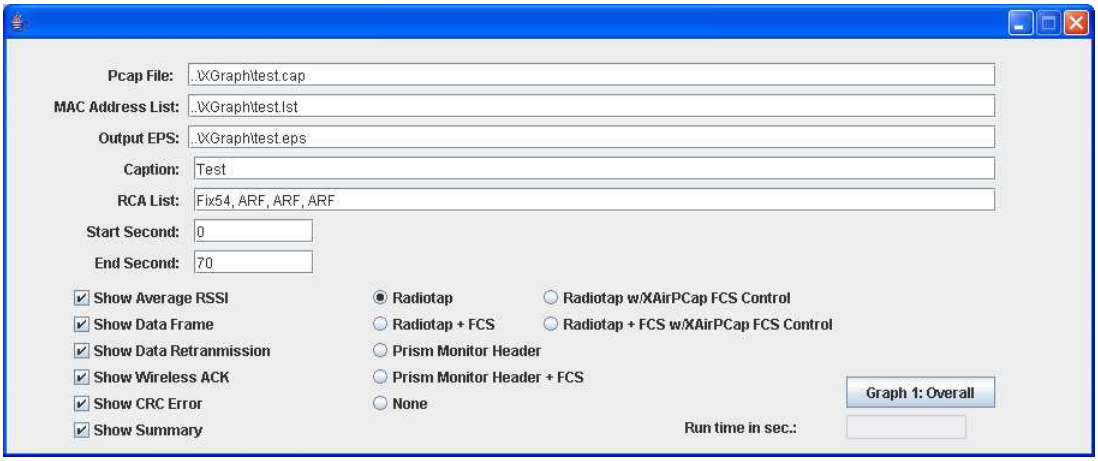


Figure 1.7: XGraph user interface

The "XGraph" is a Java program to plot the captured IEEE 802.11 traffic for analysis. It works with different status header, such as "libpcap" file, and the CRC field to verify packet status. Selectable output options and counter summary. The user interface is as Figure 1.7.

The output file is in standard EPS format which is high quality graphic and can be imported into various document type. It provides multiple views on network events and status in same time line. There are five graphs and a numerical summary at the end. They are user selectable for different requirement. The sample output is as Figure 1.8 in Page 17. The description of six portions are:

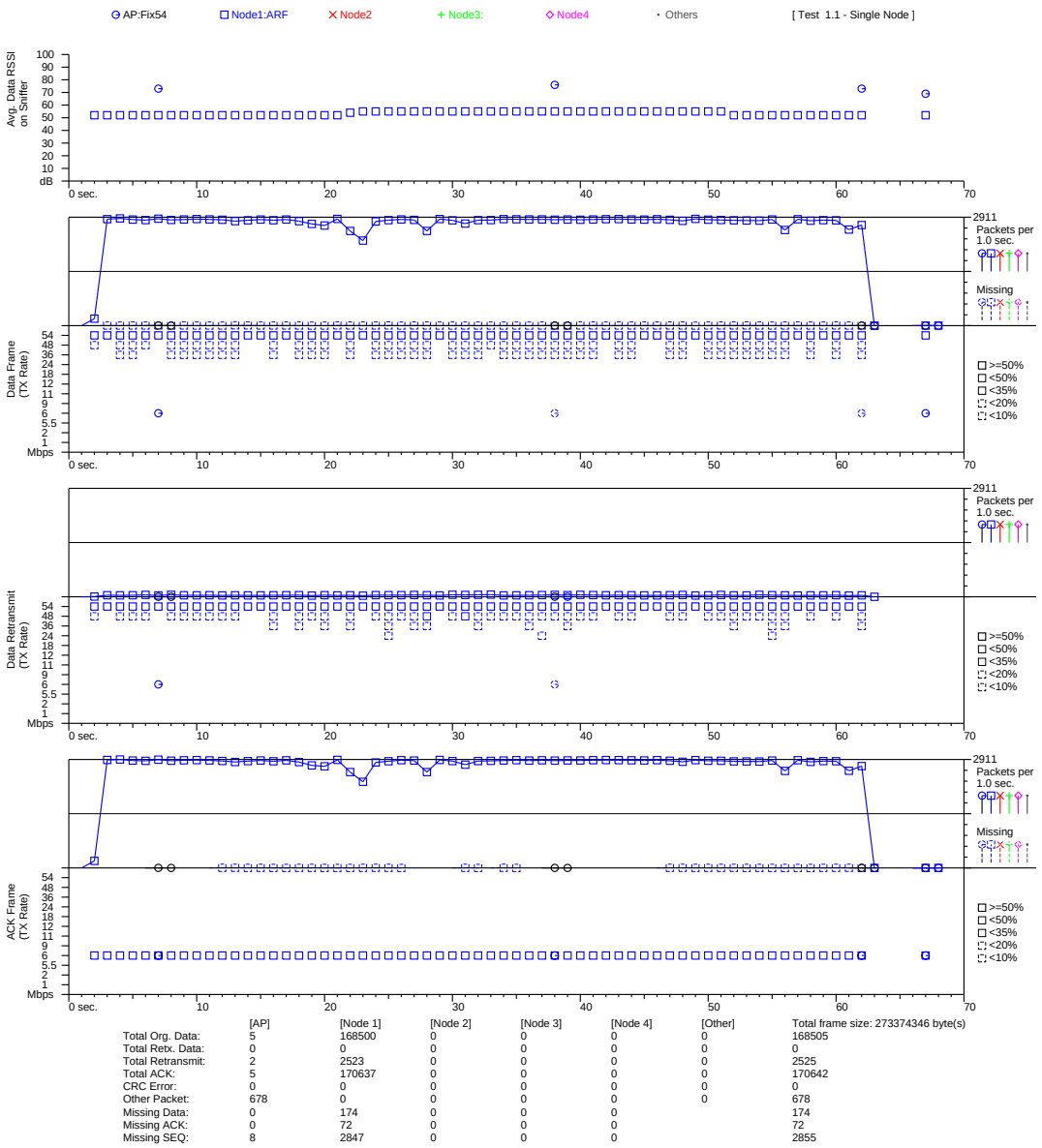
1. Average RSSI on Sniffer — This graph shows average Received Signal Strength Indication for each client / node. The unit is 0 to 100 dB. Also, there may have small gray dot which represents the detection of other network activities. The actual value for over 100 dB will be shown on top of the graph.
2. Data Frame — There are three kind of information shown in upper part of the graph.
   - Solid line — Packet count in the period.
   - Dash line — Detected missing packet count in the period.
   - Node's legend — Transmission rate is being used. The thickness of symbol represents the relative distribution between different rate used in the period. The line type shows the counter in percentage as defined on right side of the graph.

   The lower part is used to show the transmission rate detected. The detail is in Page 18 [How to read the transmission rate in the XGraph].
3. Data Retransmit — There are two kind of information shown in upper part of the graph.
   - Solid line — Packet count in the period.
   - Node's legend — Transmission rate is being used, same definition as Data Frame Graph.

   The lower part is used to show the transmission rate detected. The detail is in Page 18 [How to read the transmission rate in the XGraph].
4. ACK Frame — Same presentation style as Data Frame for 802.11 ACK frame.

5. CRC Error — Any packet with FCS error detected will be shown in this graph.

6. Summary — The table shows the total counter for each type of packet.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 273374346 byte(s)
Total Org. Data:	5	168500	0	0	0	0	168505
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	2	2523	0	0	0	0	2525
Total ACK:	5	170637	0	0	0	0	170642
CRC Error:	0	0	0	0	0	0	0
Other Packet:	678	0	0	0	0	0	678
Missing Data:	0	174	0	0	0		174
Missing ACK:	0	72	0	0	0		72
Missing SEQ:	8	2847	0	0	0		2855

Figure 1.8: Sample Output from XGraph

Missed packet detection

Due to capability of sniffer, there may have some packets to be missed in the capture processing. It is different from packet drop between device buffer and application buffer. They are not detected by the hardware device. To reduce the counting error on measurement. There are two methods to detect missing data frame count and ACK frame count. These counter can be treat as gaps in capturing.

1. Missed Data Count — Trace all captured traffic on each node. Missed Data is confirmed for two or more consecutive 802.11 ACK frame are counted.
2. Missed ACK Count — Trace all captured traffic on each node. Missed ACK is confirmed it there is no 802.11 ACK is before Data frame is counted.

Another method being used is Sequence Number Tracing. The standard 802.11 sequence number is in the range from 0 to 4095 in circularity. This is more realistic to represent the missed packet count.

How to read the transmission rate in the XGraph

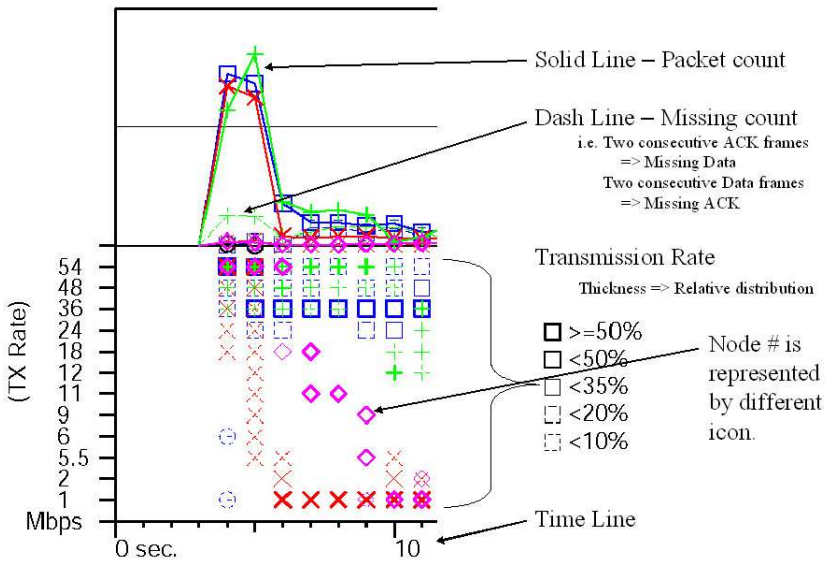


Figure 1.9: Traffic Trace from XGraph

The upper part of the graph is described in Page 16 [2. Data Frame]. The lower part of the graph is used to describe the transmission rate applied. They are shown in relative proportion in >=50%, <50%, <35%, <20%, and <10%. They are represented by line thickness and line type for the time segment. Example of combination:

1. Only one ">=50%" icon is appeared in 54Mbps — All packets are transmitted in 54Mbps at that period.

2. One ">=50%" icon on 48Mbps and one "<20%" icon on 54Mbps are appeared — It means that over 80% of packets are transmitted in 48Mbps and less than 20% transmission in 54Mbps.

3. One ">=50%" on 24Mbps, one "<20%" on 36Mbps and one "<10%" on 11Mbps — It means that over 70% of packets are transmitted in 24Mbps, less than 20% transmission in 36Mbps and less than 10% transmission in 11Mbps.

4. One "<35%" on 54Mbps, One "<35%" on 24Mbps, one "<20%" on 36Mbps and one "<10%" on 11Mbps — It means that no more than 35% are transmitted in 54Mbps, less than 35% of transmission in 24Mbps, less than 20% in 36Mbps and less than 10% in 11Mbps.

1.6 Objective and Progression

To illustrate the efficiency of different types of sniffing methodology. Propose effective testbed setup, including hardware and software, for wireless measurement.

Based on the RCA performance analysis, discuss their efficiency against network population. Introduce the Auto-Selecting the Rate Adaptation Algorithm (ASRAA) which is new on rate control. With the "Auto-RCA-Select" engine, multiple RCA logic can be selected for specific application or situation.

Finally, discussion of future work and improvement will also be presented at the end of this report.

Chapter 2

Testbed Design

There are three testbeds to be used for "Sniffer Comparison", "RCA Performance Analysis" and "Auto-Selecting the Rate Adaptation Algorithm". All testbeds will be controlled by the new general tool RemoteCall.

2.1 Test Packet and Channel Capacity

I use UDP packet in all test. This eliminates any TCP based contention control. All analysis is focused on 802.11 traffic only. The detail of test packet is defined as follows.

Packet Contain	Size (byte)
802.11 MAC Header (3xAddress w/QoS)	26
LLC + SNAP	8
IPv4 Header	20
UDP Header	8
UDP Data	1470
802.11 FCS	4

Table 2.1: 802.11 Data Frame with Encapsulated UDP Packet

Packet Contain	Size (byte)
802.11 ACK	10
802.11 FCS	4

Table 2.2: 802.11 ACK Frame

There are two major frame types to be filled up the channel. They are Data Frame from the Sender and corresponded ACK Frame from the AP. Test will be run on 802.11g only environment, i.e. no consideration of 802.11b compatibility. No RTS / CTS will be used. The beacon interval is set to 100ms. The transmission time for each UDP packet and ACK pair is calculated as following, the details are shown in Table 2.1 and Table 2.2.

$$DIFS + Preamble + Data\ Frame + SIFS + Preamble + ACK\ Frame$$

$$= 28 + 20 + 234 + 10 + 20 + 10$$

$$= 322\ \mu s$$

The theoretical maximum throughput in application layer will be,

$$1 / Packet\ Transmission\ Time\ in\ \mu s * Data\ Payload\ Size\ in\ bits / 1000000$$

$$= 1 / 322\ \mu s * 1470 * 8 / 1000000$$

$$= 36.52Mbps$$

The UDP payload is set to 1470 which is used to prevent IP fragmentation for packet size over the usual MTU limitation for 1500 bytes. With consideration of 5% overhead for any other 802.11 Control and Management Frame, the application layer output stream is set to 35Mbps which can fill up the channel in all followed test.

There is a short test to verify above setting. Three computers are used to send out UDP packet in 50Mbps. Their hardware configuration are shown in Table 2.3. Test is done individually. The result in Table 2.4 shows that the setting of 35Mbps is appropriate for sender to fill up channel in all test below.

Node	Hardware	NIC
AP	AMD Sempron 2400+ 1.67GHz, 512M RAM	MadWiFi
Computer 1	Intel Pentium M 1.40GHz, 768M RAM	MadWiFi
Computer 2	Intel Pentium M 1.40GHz, 768M RAM	MadWiFi
Computer 3	Intel Pentium III 500MHz, 320M RAM	MadWiFi

Table 2.3: Configuration for Goodput Test

Test	Computer 1	Computer 2	Computer 3
1	35.2	35.1	34.8
2	35.3	35.2	34.8
3	35.3	35.1	34.8
4	35.3	35.0	34.9
5	35.3	35.2	34.9

Table 2.4: Recorded Goodput in Mbps

2.2 Testbed for Sniffer Comparison

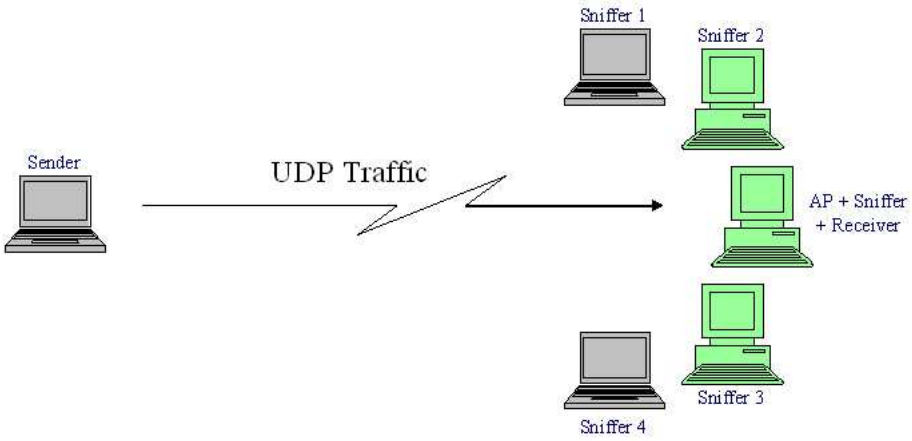


Figure 2.1: Testbed Setup for Sniffer Comparison

The Figure 2.1 shows the testbed setup. There are one Sender, one AP + Receiver + Sniffer, and four individual Sniffers. The Sender is located within 1 meter from the AP and all Sniffer's antenna are placed very close to the AP's antenna. The Sender will generate UDP traffic to full load the 802.11g channel for 60 seconds by using the following command:

iperf -c 192.168.x.x -p 200x -u -l 1470 -b 35M -t 60

2.2.1 Hardware Configuration

Refer to the configuration in Table 2.5, the description of component are:

- The "Sender" is the traffic generator.
- The "AP+Receiver+Sniffer" is a multi-role node. It provides AP function, data receiver, and internal sniffer in a single machine. This will eliminate the propagated delay in the case of separated AP and Receiver.
- The "Sniffer 1" is consist of a normal MadWiFi compatible NIC which will operate in monitor mode. It is enforced to be lower performance sniffer caused by the lower grade processor. It is used to be the reference of hardware efficiency effect.

Node	Hardware	Sniffer	OS
Sender	Intel Pentium M 1.40GHz, 768M DDR-RAM		Linux FC6
AP + Receiver + Sniffer	AMD Sempron 2400+ 1.67GHz, 512M DDR-RAM	MadWiFi	Linux FC6
Sniffer 1	Intel Pentium III 500MHz, 320M SD-RAM	MadWiFi	Linux FC6
Sniffer 2	AMD Athlon XP 2100+ 1.73GHz, 512M DDR-RAM	MadWiFi	Linux FC6
Sniffer 4	Intel Pentium M 1.40GHz, 768M DDR-RAM	AirPcap(a) MadWiFi	Windows XP Linux FC6
Sniffer 3	Intel Core2 Quad Q6600 3.24GHz, 2G DDR-RAM	AirPcap(b) MadWiFi	Windows XP Linux FC6

Table 2.5: Nodes Configuration for Sniffer Test

- The "Sniffer 2" is also consist of a MadWiFi compatible NIC in monitor mode. The hardware platform is very similar to the "AP+Receiver+Sniffer". It is used to be the verifier for different between Internal and External Sniffer.
- The "Sniffer 3" is same hardware performance as the "Sender". It uses the AirPcap(a) model:BELKIN F5D7050 passive capturer in Windows platform for different sniffer type comparison. In the other case, it use MadWiFi in Linux platform for same sniffer type comparison.
- The "Sniffer 4" is a high performance Windows based machine with a dedicated passive capturer, the AirPcap(b) model:HWU54G, for different sniffer type comparison, i.e. AirPcap vs. MadWiFi. Also, it will be deployed with MadWiFi in Linux platform for hardware efficiency test of same sniffer type comparison.

2.2.2 Standard Tools

iperf

All test data are generated by the "iperf" version 2.0.2 [2] which is a popular throughput measurement tool. It can be worked as client (sender) or server (receiver). It supports TCP and UDP test. It can control test duration, buffer size and etc. Please refer to man page for details.

tshark

The command line tool in "Wireshark" is used for packet checkup and verification. It is higher efficiency than the GUI tool. It is common tool for packet capture and analysis. This version 0.99.5 is come with AirPcap and customized to show rich ratio information. The "Wireshark" GUI is easy to use and very good on rapid trace. It provides many simple graph and packet filtering for specific need of analysis.

2.2.3 Custom Tools

Auto-Selecting the Rate Adaptation Algorithm

The "Auto-Selecting the Rate Adaptation Algorithm" (ASRAA) is a custom RCA logic which is implemented in MadWiFi. It provides the mechanism for RCA selection in run-time. The /proc standard file system in Linux, which is the root of hierarchical register for system and device information. It is used for run-time parameter exchange to control the operation.

XGraph

The "XGraph" is a Java program to plot the captured result for analysis. It works with different status header and the CRC field to verify packet status. It provides multiple views on network events and status in same time line. Selectable output options and counter summary.

The output file is in standard EPS format which is high quality graphic and can be imported into various document type.

2.2.4 Test Case

There three test cases to be defined. They are,

Node	Hardware	Sniffer	OS
Sender	Intel Pentium M 1.40GHz		Linux FC6
AP + Receiver + Sniffer	AMD Sempron 2400+ 1.67GHz	MadWiFi	Linux FC6
Sniffer 1	Intel Pentium III 500MHz	MadWiFi	Linux FC6
Sniffer 2	AMD Athlon XP 2100+ 1.73GHz	MadWiFi	Linux FC6
Sniffer 3	Intel Pentium M 1.40GHz	AirPcap(a)	Windows XP
Sniffer 4	Intel Core2 Quad 3.24GHz	AirPcap(b)	Windows XP

Table 2.6: Setup for Sniffer Test Case 1 and 2

1. Passive Capturer vs. Normal NIC in Monitor Mode — There is a lower cost passive capturer in the market. The "AirPcap" is USB 2.0 device and to be worked in Windows platform only. The manufacturer claims that the device can capture any packet like from the air, even the CRC check is failed. In this test group, it will be compared with well known MadWiFi device in monitor mode. The MadWiFi is used widely in wireless network measurement. This test will illustrate the pros and cons of their own character. The hardware setup is as Table 2.6.

2. Various Setup Effect — Beside of default setting for those device in above test as Table 2.6. AirPcap provides an unique run-time parameters for performance tunning, adjustable receiver buffer size. The buffer size will be changed to verify the enhancement. This is not applied for MadWiFi, it can be changed in compile time only.

3. Hardware Effect — This test is used to check out how fast machine to be appropriate for high volume traffic capturing. The hardware performance different is shown in Table 2.7.

Node	Hardware	Sniffer	OS
Sender	Intel Pentium M 1.40GHz		Linux FC6
AP + Receiver + Sniffer	AMD Sempron 2400+ 1.67GHz	MadWiFi	Linux FC6
Sniffer 1	Intel Pentium III 500MHz	MadWiFi	Linux FC6
Sniffer 2	AMD Athlon XP 2100+ 1.73GHz	MadWiFi	Linux FC6
Sniffer 3	Intel Pentium M 1.40GHz	MadWiFi	Linux FC6
Sniffer 4	Intel Core2 Quad 3.24GHz	MadWiFi	Linux FC6

Table 2.7: Setup for Sniffer Test Case 3

2.2.5 Scope of Test

All of above three tests are applied for Internal and External Sniffer only. The Virtual Sniffer is local loop based, some kind of hardware based transmission will not be detected. Such as ACK frame generated by Hardware Abstraction Layer in MadWiFi cannot be captured. Also, the key task for Multiple Sniffers is post data processing on merging and filtering. It is software engineering intensive. Both two type of sniffer are not interested on this study.

2.3 Testbed for RCA Performance Analysis

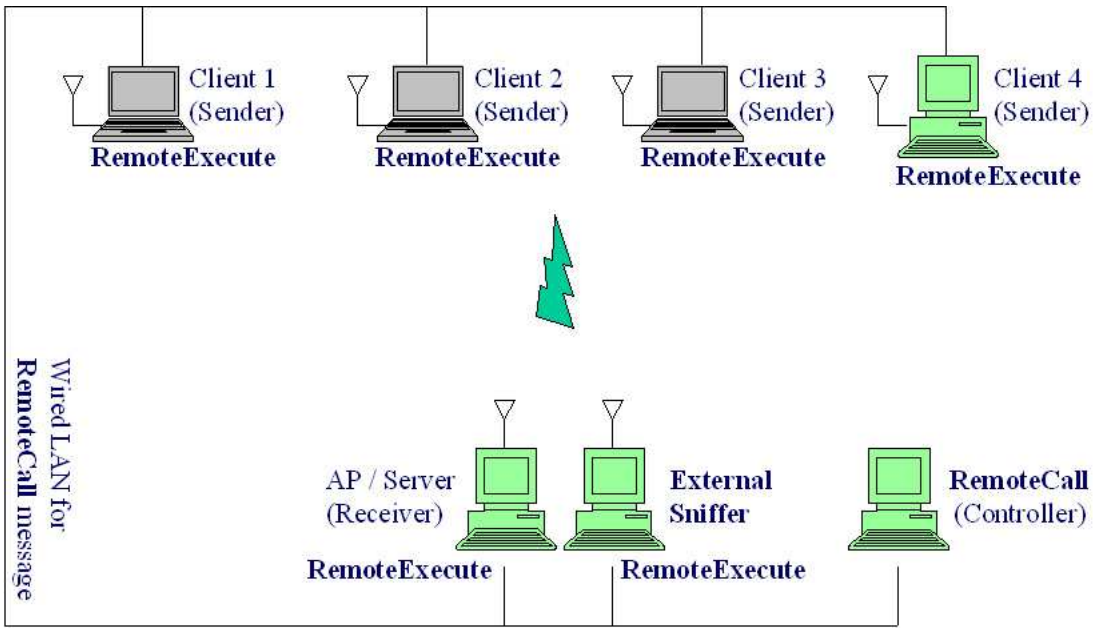


Figure 2.2: Testbed Setup for RCA Performance Comparison

The Figure 2.2 shows the component setup for RCA Performance Comparison against network population. There are four Clients, one AP / Server, and one Sniffer. All Clients are located 1.5m away from the AP. The Sniffer's antenna is placed on very close to AP's antenna.

The wireless network is based on 802.11g only infrastructure mode. No RTS / CTS is used. All Clients and the AP / Server are controlled by a RemoteCall via the 100BaseT wired network. The wired interconnections are through a switch / hub for task synchronization across multiple nodes. This will isolate all test related control messages from measured wireless network for accurate result.

2.3.1 Hardware Configuration

All nodes configuration are shown in Page 30 Table 2.8.

Node	Hardware	O.S.
Client 1	Intel Pentium M 1.4 GHz, 768M DDR-RAM, Atheros 5212 PCMCIA NIC	CentOS 5.1, Linux Kernel 2.6.18-53.el5
Client 2	Intel Pentium M 1.4 GHz, 768M DDR-RAM, Atheros 5212 PCMCIA NIC	CentOS 5.1, Linux Kernel 2.6.18-53.el5
Client 3	Intel Pentium III 500 MHz, 320M SD-RAM, Atheros 5212 PCMCIA NIC	CentOS 5.1, Linux Kernel 2.6.18-53.el5
Client 4	Intel Core2 Quad 3.24GHz, 2G DDR2-RAM, Atheros 5212 PCI NIC	CentOS 5.1, Linux Kernel 2.6.18-53.el5
AP + Server	AMD Sempron 2400+ 1.67GHz, 512M DDR-RAM, Atheros 5212 PCI NIC	CentOS 5.1, Linux Kernel 2.6.18-53.el5
Sniffer	AMD Athlon XP 2100+ 1.73GHz, 512M DDR-RAM, Atheros 5212 PCI NIC	CentOS 5.1, Linux Kernel 2.6.18-53.el5
Controller	Intel Pentium 4 3.0 GHz, 512M DDR-RAM	Window XP SP1

Table 2.8: Nodes Configuration for RCA Comparison

Refer to above configuration Table 2.8, the component selection are:

- The Client 1 and 2 are identical in hardware and software. It is assumed to be fair in RCA comparison, no evident advantage will be taken by either one. Pure RCA performance is expected in the measurement.
- The Client 3 is enforced to be lower performance than other clients which is caused by the lower grade central processing unit. In certain identical RCA test, it is used to illustrate the effect of hardware performance.
- The Client 4 is PC based in stead of notebook platform.
- The AP / Server is a multi-role node. It provides AP function and server service in a single machine. This will eliminate the propagated delay in the case of separated AP and Server.
- The Sniffer, which is recommended in above Sniffing Methodology Comparison, is located near by the AP / Server. Their antenna are put on side by side. The sniffing device is the MadWiFi in monitor mode. The election of this type of sniffer is shown on following test result. It is high efficiency setup to capture high volume data traffic.

- The Controller is used as centralized remote control to all other node in the testbed through the wired network.

2.3.2 Standard Tools

iperf

Once again, all test data are generated by the "iperf" version 2.0.2 [2] which is a popular throughput measurement tool. It will be used as traffic generator, i.e. stream out UDP packet in 35Mbps to fill up the channel.

tshark

The command line tool in "Wireshark" is used for packet capturing. It is higher efficiency than the GUI tool. It is compatible to MadWiFi and AirPcap. The standard RadioTap header is used for extra radio information, such as single strength and CRC value. Those information can be used for packet verification.

2.3.3 Custom Tools

There are three new tools are developed for RCA selection, remote control and data presentation. These tools are used to produce a consolidated output which covers multiple correlated factors. All data are aligned to same time line for easy reading. The numerical summary of each test is also included on the output for further checking.

Auto-Selecting the Rate Adaptation Algorithm

The Auto-Selecting the Rate Adaptation Algorithm (ASRAA) is a custom RCA logic which is implemented in MadWiFi. It provides the mechanism for RCA selection in run-time. It uses the /proc standard file system in Linux, which is the root of hierarchical register for system and device information, for run-time parameter exchange to control the operation.

RemoteCall

The "RemoteCall" and "RemoteCallExecute" are used for remote control and task scheduling. All control messages are sent through IP based 100BaseT wire network. It is separated from the object wireless media, this eliminates the interference which is caused by remote control signal. Remote control commands are sent via TCP unicast for single host setup. The trigger message is sent via UDP broadcast to signal all clients to run the test. UDP is a lower latency protocol, no connection overhead as TCP. It is the effective signaling for simple local area network, i.e. no routing is needed for peer communication.

XGraph

The "XGraph" is used to generate high quality graphic in standard EPS format which can be imported into various document type. It provides the counter summary to describe the traffic status.

2.3.4 Test Case

There are three types of test case to be performed:

1. Identical RCA in all Clients — Check RCA behavior on different hardware performance. Is higher power machine always be the dominator ?
2. Uncommon RCA against popular RCA — Popular RCA, such as ARF and AMRR, are widely implemented. They are usually be simple and effective. For those relative new and complex RCA, ONOE and SAMPLE may or may not be over performance than popular algorithms. This type of test is used to explore their competitiveness.
3. Cross comparison — Every node use different RCA. Trace their reaction for network events. Find out who is the advancer in a challenging network.

Based on above categories, the test cases shown in Table 2.9 are planned to be performed. The aim of these tests is to find out the efficiency of different RCA and the general behavior

on challenging environment.

Type	Test	Client 1	Client 2	Client 3	Client 4
1	1	ARF	ARF	ARF	ARF
	2	AMRR	AMRR	AMRR	AMRR
	3	ONOE	ONOE	ONOE	ONOE
	4	Sample	Sample	Sample	Sample
2	1	ONOE	ARF	ARF	ARF
	2	Sample	ARF	ARF	ARF
	3	ONOE	AMRR	AMRR	AMRR
	4	Sample	AMRR	AMRR	AMRR
3	1	ARF	AMRR	ONOE	Sample
	2	ONOE	Sample	ARF	AMRR
	3	Sample	ONOE	AMRR	ARF
	4	AMRR	ARF	Sample	ONOE

Table 2.9: Test for RCA Comparison

2.3.5 General Setup for All RCA Test

There is a standard setup for all of above 12 test case. The RCA for AP is set to "Fix" and transmission rate is fixed on 54M, no dynamic rate control algorithm to be applied. It makes the AP to react any clients request as soon as possible. All tests are done through UDP transmission. It is simple, no complex congestion control as TCP. There is no return packet (TCP ACK) from AP to affect the measurement.

The RCA used in clients is according to test case listed on Table 2.9. All clients will send out packets continuously for 60 seconds. The payload size is fixed as the "iperf" default UDP server buffer size, 1470 bytes. The command line is,

iperf -c 192.168.x.x -p 200x -u -l 1470 -b 35M -t 60

The setting criteria is discussed in Section 2.1. Also, there are only four type of packet are expected in the wireless network.

1. Data — UDP packet from clients to server.
2. 802.11 ACK — Acknowledgement from AP for good packet.
3. Error — Any packet with bad CRC or belong to another network in same ratio channel.
4. Other — Any other 802.11 control and management frame, such as Beacon and Association / De-association.

Above four type of frame will be traced during RCA characterization.

2.4 Testbed for Auto-Selecting the Rate Adaptation Algorithm

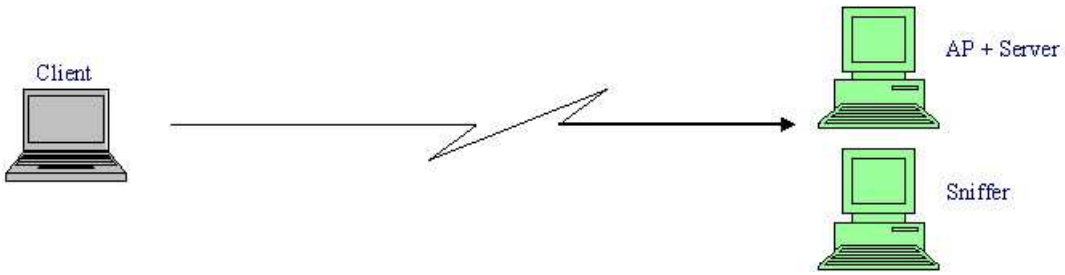


Figure 2.3: Testbed Setup for Auto-Selecting the Rate Adaptation Algorithm

The testbed setup is shown in Figure 2.3. There are one Client, one AP + Server, and one individual Sniffer. The Client is located within 1 meter from the AP and the Sniffer antenna is placed very close to the AP's antenna. The Client will generate traffic by using different application. The test will try out the auto RCA selection. The selection is policy based. Currently, the policy shows in Table 2.10 which is set for testing only. It is not proved. The well mating is not in this study.

Policy	Application	RCA	Remark
1	FTP	ARF	Large packet, large volume
2	Video Streaming	AMRR	Small packet, constant interval
3	Other	Sample	General network traffic

Table 2.10: Policy for ASRAA Test

2.4.1 Hardware Configuration

Refer to the configuration Page 36 Table 2.11, the description of component are:

- The "Client" will generate the traffic by using different application, such as FTP or Video Stream.

- The “AP + Server” is a multi-role node. It provides AP function and application server in a single machine. This will eliminate the propagated delay in the case of separated AP and Server.
- The Sniffer, which is recommended in above Sniffing Methodology Comparison, is located near by the AP / Server. Their antenna are put on side by side. The sniffing device is the MadWiFi in monitor mode. The election of this type of sniffer is shown on following test result. It is high efficiency setup to capture high volume data traffic.

Node	Hardware	Sniffer	OS
Client	Intel Pentium M 1.40GHz, 768M DDR-RAM		Linux FC6
AP + Server	AMD Sempron 2400+ 1.67GHz, 512M DDR-RAM	MadWiFi	Linux FC6
Sniffer	AMD Athlon XP 2100+ 1.73GHz, 512M DDR-RAM	MadWiFi	Linux FC6

Table 2.11: Nodes Configuration for ASRAA

2.4.2 Standard Tools

FTP

FTP is used to test policy based auto RCA selection in ASRAA implementation. The FTP server is “proftpd 1.3.0a” and the client is “gftp 2.0.18-3.2.2”. Both program is commonly used in Linux platform.

Video Streaming

The “vlc 0.8.5-6” is multifunction media player. It can be used as video player or streaming server. It is easy to use tool for video testing.

2.4.3 Test Case

In current state, the policy for auto RCA selection in ASRAA is very simple. This is initial version. It is used to test the framework only. It is not well defined. The RCA is expected to be charged according to application running.

Policy	Application is running	RCA will be used
1	FTP	ARF
2	Video Streaming	AMRR
3	Other	Sample

Table 2.12: Initial Policy for ASRAA RCA Switching

Chapter 3

Sniffer Comparison

3.1 Comparison Objective

Compare two popular sniffers and describe their characters.

1. **AirPcap** is a passive capturer from CACE Technologies. It is customized to capture ANY ratio signal in the user specified 802.11 b/g channel. The RadioTap header is included in each captured packet for analysis. Dur to USB ability, high packet loss in capturing may occur.
2. **MadWiFi Monitor Mode** is a virtual NIC. It can be used for packet capturing. The PRISM or RadioTap header for ratio information is user selectable. Only those packets with correct FCS value will be pass through the driver. The environmental noise is missed on any measurement. This is the main different from AirPcap.

3.2 Passive Capturer vs. NIC in Monitor Mode

Table 3.1 and Table 3.2 show the result of two tests, each test consist of five measurements. There is a sender which transmits UDP packet in 35Mbps to the AP. The result shows that the Internal Sniffer in the AP and External Sniffer with AirPcap (Sniffer 1, 4 and 5) are performed

Sniffer	1	**2**	3	4	5
Host	AP	**Athlon XP**	Pentium III	Pentium M	Core2 Quad
Device	MadWiFi	**MadWiFi**	MadWiFi	AirPcap(a)	AirPcap(b)
Packet Sent	% Captured	**% Captured**	% Captured	% Captured	% Captured
138721	94.1(94.3)	**96.8(97.3)**	88.0(86.3)	97.2(0.4)	98.6(100)
138715	92.4(90.9)	**96.4(97.1)**	87.3(84.3)	96.2(0.3)	98.9(100)
138718	93.2(92.7)	**96.6(97.4)**	88.6(85.4)	95.5(0.5)	98.8(100)
138743	92.8(92.7)	**96.8(97.5)**	87.9(86.3)	96.7(0.4)	98.5(100)
138814	93.9(94.0)	**97.1(97.8)**	87.8(86.2)	97.6(0.3)	98.5(100)

Table 3.1: Sniffer Test 1 - Percentage of Data (ACK) Captured

Sniffer	1	**2**	3	4	5
Host	AP	**Athlon XP**	Pentium III	Pentium M	Core2 Quad
Device	MadWiFi	**MadWiFi**	MadWiFi	AirPcap(a)	AirPcap(b)
Packet Sent	% Captured	**% Captured**	% Captured	% Captured	% Captured
178478	32.0(76.4)	**99.1(99.8)**	95.2(92.1)	77.8(86.1)	73.2(89.4)
178504	36.1(78.5)	**99.2(99.9)**	95.3(92.3)	78.2(86.2)	65.7(87.8)
178504	29.5(75.5)	**99.2(99.8)**	95.4(92.3)	71.4(81.6)	64.5(89.2)
178526	24.1(41.4)	**98.8(100)**	95.8(92.8)	34.9(51.9)	86.4(90.1)
178516	26.5(82.6)	**98.9(99.9)**	94.9(91.8)	44.3(29.5)	90.3(96.2)

Table 3.2: Sniffer Test 2 - Percentage of Data (ACK) Captured

unreliable. For both External Sniffer with MadWiFi (Sniffer 2 and 3) are worked stable but the performance is seem relative to the processor speed. The Sniffer 2 (Athlon XP) can capture over 96% of network traffic all the time. The summarized comments are:

1. MadWiFi in monitor mode as a dedicated sniffer is performed well.

2. Processor speed should be concerned for sniffing in high volume network traffic.

3.3 Effect of Hardware Platform

Sniffer	**1**	2	**3**	4
Host Type	**PC**	Notebook	**Notebook**	PC
CPU	**Athlon XP**	Pentium III	**Pentium M**	Core2 Quad
Device	**MadWiFi**	MadWiFi	**MadWiFi**	MadWiFi
Packet Sent	**% Captured**	% Captured	**% Captured**	% Captured
178406	**97.8(100)**	96.2(94.0)	**97.5(98.4)**	97.1(98.6)
178512	**88.1(100)**	93.7(90.6)	**97.9(98.9)**	97.2(98.6)
178494	**98.8(100)**	94.5(90.9)	**95.7(96.1)**	95.0(96.0)
178102	**96.3(100)**	94.3(90.8)	**98.9(99.6)**	90.9(92.1)
178520	**98.5(100)**	95.7(92.4)	**98.1(98.6)**	95.4(96.4)

Table 3.3: Sniffer Test 3 - Percentage of Data (ACK) Captured

Refer to last section, the "MadWiFi in monitor mode" is selected to be the appropriate sniffing device. This test is used to check the sniffing performance to be affected by the host ability. The result in Table 3.3 shows that multi-core processor (Sniffer 4) does not have much advantage on network traffic sniffing, it seems a single task processing. Both sniffing machine with reasonable speed, the Athlon XP and Pentium M (Sniffer 1 and 3), can finish the job.

3.4 Selected Sniffer for Traffic Capturing

Based on above comparisons, the Athlon XP based "MadWiFi in monitor mode" is the appropriate sniffer for all other test in this thesis. Its capturing ability is reliable, over 95% in all test. Invalid packets are not interested in designed test, even MadWiFi cannot capture environmental noise.

Chapter 4

RCA Performance Analysis

4.1 Study Objective

The objective of the comparison is to check out various RCA's behavior against network population under different scenarios, such as all clients using the same RCA or there are different algorithm in some nodes. Cross comparison will be done for characterization. There are total four RCAs to be used in comparison. They are ARF [7], AMRR [8], ONOE [3], and Sample [6].

UDP traffic is used for all test that eliminates the TCP congestion control to affect the measurement. The AP is expected to send out IEEE 802.11 management and ACK frame. Only the clients side competition is interested on the analysis.

All XGraph output in this chapter are simplified to show data packet count only, please refer to Appendix B for more detail figure.

4.2 Type 1 - Identical RCA in all nodes

These tests are used to observes the population effect in wireless network. Check the overall efficacy when number of active node is increased. Also, the fairness of multiple active nodes is concerned.

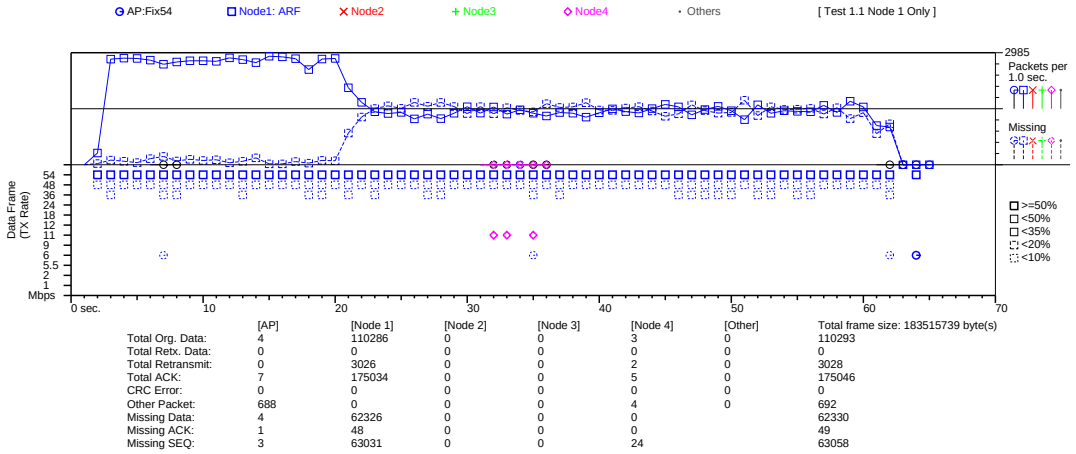
4.2.1 Test 1.1 - All clients use ARF

No. of Active Node	Client 1	Client 2	Client 3	Client 4	Total
1	34.3				Individual
		34.5			Individual
			26.7		Individual
				28.4	Individual
2	17.0	17.0			34.0 Mbps
3	9.6	8.3	9.3		27.2 Mbps
4	620K	491K	179K	324K	1.6 Mbps

Table 4.1: Test 1.1 - ARF - iperf server report

This is to test the performance of ARF relative to the population. The result shows that the total goodput for single and two active nodes are around 34 Mbps which is close to the 35 Mbps reference channel bandwidth. But, the aggregated utilization is inefficient when three nodes are actively sending packets, it is more serious for four active nodes. This shows that ARF is not well performed in busy environment. The more detail of network traffic record is on next pages.

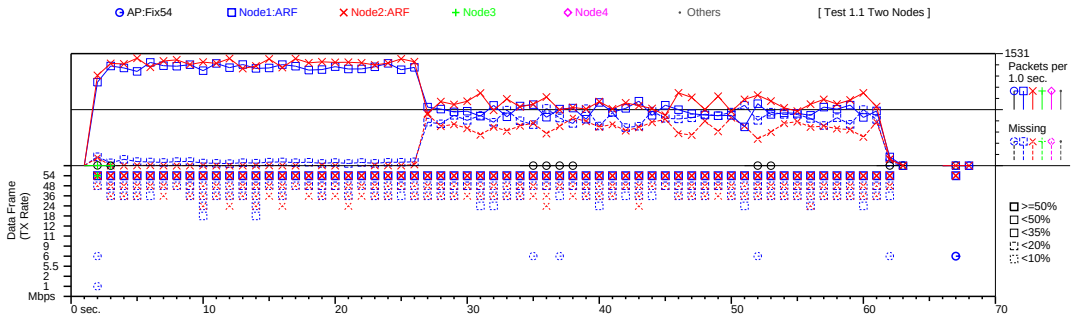
1. The single active node can use the whole channel to maximize the utilization. The ARF is keep trying to use highest transmission rate. There are few lower rate to be used.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 183515739 byte(s)
Total Org. Data:	4	110286	0	0	3	0	110293
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	0	3026	0	0	2	0	3028
Total ACK:	7	175034	0	0	5	0	175046
CRC Error:	0	0	0	0	0	0	0
Other Packet:	688	0	0	0	4	0	692
Missing Data:	4	62326	0	0	0		62330
Missing ACK:	1	48	0	0	0		49
Missing SEQ:	3	63031	0	0	24		63058

Figure 4.1: Test 1.1 - Single active client uses ARF

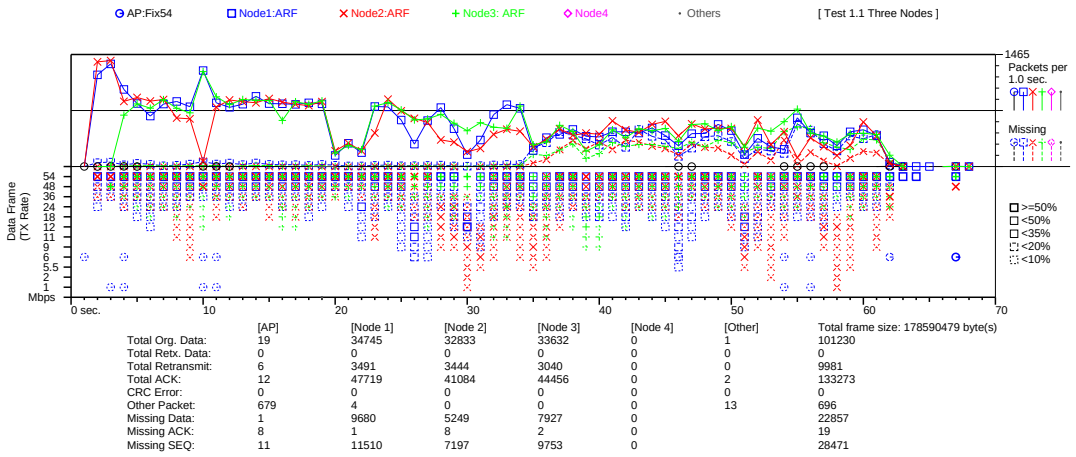
2. The fairness on channel utilization is shown in two active node case. The channel utilization is still in high level. The lower rate transmission is increased due to higher collision. But, most of transmission are still in highest rate range.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 213157782 byte(s)
Total Org. Data:	11	59644	64750	1	0	0	124406
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	3	3826	4106	0	0	0	7935
Total ACK:	8	87435	87126	1	0	0	174570
CRC Error:	0	0	0	0	0	0	0
Other Packet:	684	0	0	0	0	0	684
Missing Data:	0	24236	18859	0	0		43095
Missing ACK:	4	14	18	0	0		36
Missing SEQ:	1	24284	19032	0	0		43317

Figure 4.2: Test 1.1 - Two active clients use ARF

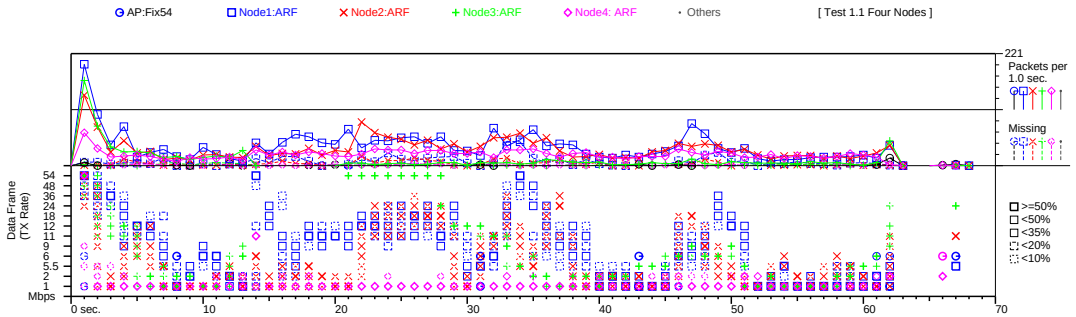
3. The fairness on channel utilization is still shown in three active node case. The total goodput is dropped. The lower rate transmission is increased a lot due to higher collision. High rate transmission cannot be kept.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 178590479 byte(s)
Total Org. Data:	19	34745	32833	33632	0	1	101230
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	6	3491	3444	3040	0	0	9981
Total ACK:	12	47719	41084	44456	0	2	133273
CRC Error:	0	0	0	0	0	0	0
Other Packet:	679	4	0	0	0	13	696
Missing Data:	1	9680	5249	7927	0		22857
Missing ACK:	8	1	8	2	0		19
Missing SEQ:	11	11510	7197	9753	0		28471

Figure 4.3: Test 1.1 - Three active clients use ARF

4. The ARF is poor performance in busy network. No one can try to use higher transmission rate. It is not adaptive on environment change.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 13731360 byte(s)
Total Org. Data:	35	2396	2078	748	1372	0	6629
Total Retx. Data:	1	0	0	1	0	0	2
Total Retransmit:	9	516	513	207	701	0	1946
Total ACK:	29	3371	2722	1004	1670	0	8796
CRC Error:	0	0	0	0	0	0	0
Other Packet:	671	1	1	7	1	0	681
Missing Data:	7	525	255	75	60		922
Missing ACK:	13	6	8	2	199		228
Missing SEQ:	27	470	218	72	131		918

Figure 4.4: Test 1.1 - Four active clients use ARF

4.2.2 Test 1.2 - All clients use AMRR

No. of Active Node	Client 1	Client 2	Client 3	Client 4	Total
1	32.2				Individual
		31.5			Individual
			25.1		Individual
				20.4	Individual
2	15.8	15.4			31.2 Mbps
3	12.0	10.8	1.1		23.9 Mbps
4	5.8	5.9	541K	4.3	16.5 Mbps

Table 4.2: Test 1.2 - AMRR - iperf server report

This is to test the performance of AMRR relative to the population. The result shows that the total goodput for single and two active nodes are around 31 Mbps which is close to the 35 Mbps reference channel bandwidth. The aggregated utilization is efficient then ARF when over three nodes are actively sending packets. The performance drop is not too serious as ARF case in high population.

1. The single active node can use the whole channel to maximize the utilization. The AMRR can keep to use highest transmission rate. No lower rate is used for data send.

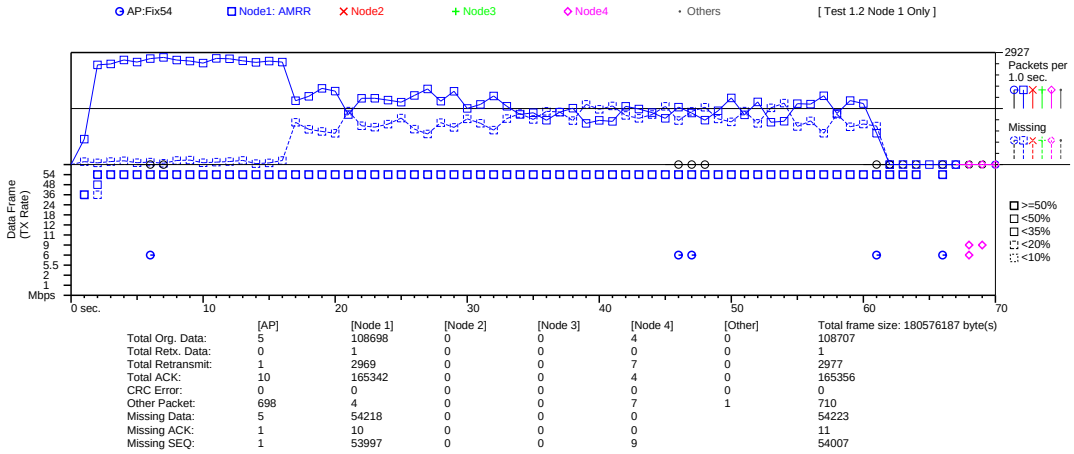
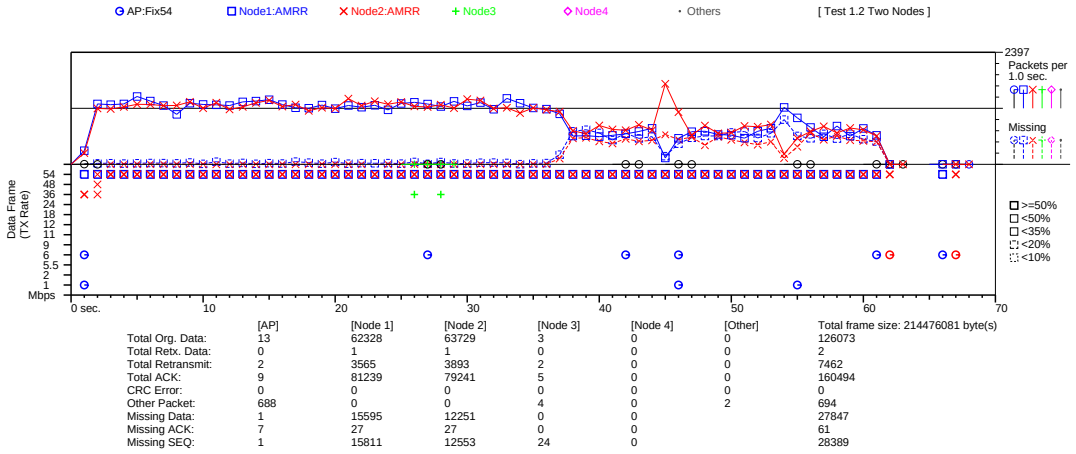


Figure 4.5: Test 1.2 - Single active client uses AMRR

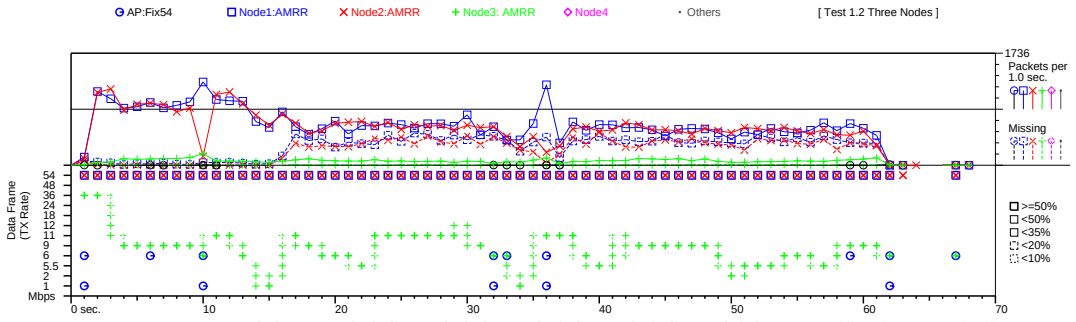
2. Similar to ARF, the fairness on channel utilization is shown in two active node case. The channel utilization is still in high level. The lower rate transmission is increased due to higher collision. But, most of transmission are still in highest rate range. The drop of packet count at the end of graph seems to be caused by the ability of the sniffer.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 214476081 byte(s)
Total Org. Data:	13	62328	63729	3	0	0	126073
Total Retx. Data:	0	1	1	0	0	0	2
Total Retransmit:	2	3565	3893	2	0	0	7462
Total ACK:	9	81239	79241	5	0	0	160494
CRC Error:	0	0	0	0	0	0	0
Other Packet:	688	0	0	4	0	2	694
Missing Data:	1	15595	12251	0	0		27847
Missing ACK:	7	27	27	0	0		61
Missing SEQ:	1	15811	12553	24	0		28389

Figure 4.6: Test 1.2 - Two active clients use AMRR

3. The fairness on channel utilization is not shown in three active node case. The total goodput is still kept in high level. The Node 3 is not performed well, the transmission rate is lower than others. It may due to weak hardware platform.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 145279532 byte(s)
Total Org. Data:	25	40137	37562	4396	0	0	82120
Total Retx. Data:	0	1	0	0	0	0	1
Total Retransmit:	5	3378	3340	1265	0	0	7988
Total ACK:	22	61265	55991	5821	0	0	123099
CRC Error:	0	0	0	0	0	0	0
Other Packet:	694	0	6	0	0	0	700
Missing Data:	3	17915	15409	341	0		33668
Missing ACK:	8	12	21	0	0		41
Missing SEQ:	6	18459	15934	358	0		34757

Figure 4.7: Test 1.2 - Three active clients use AMRR

4. Similar to three active nodes case, the fairness on channel utilization is not shown. Even the total goodput is dropped. High rate transmission is not for Node 3. The more detail of network traffic record is on Figure 4.8.

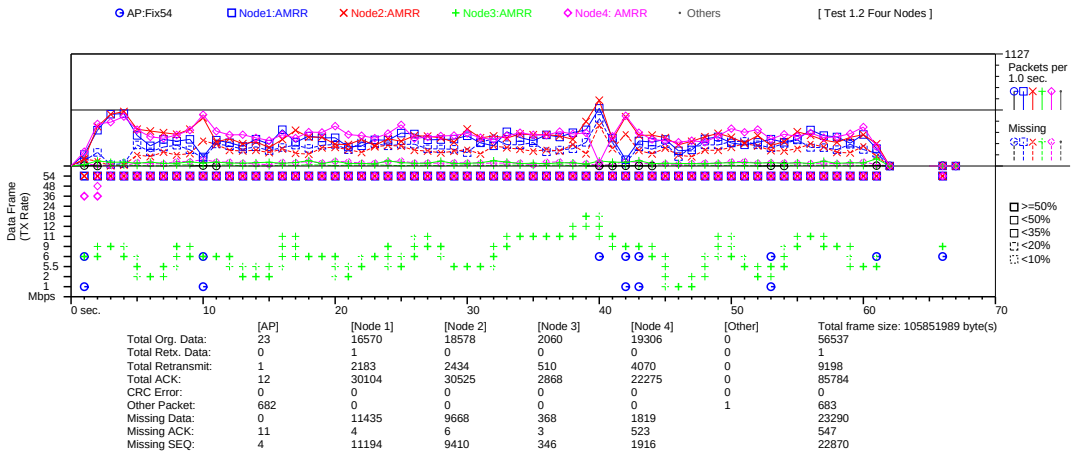


Figure 4.8: Test 1.2 - Four active clients use AMRR

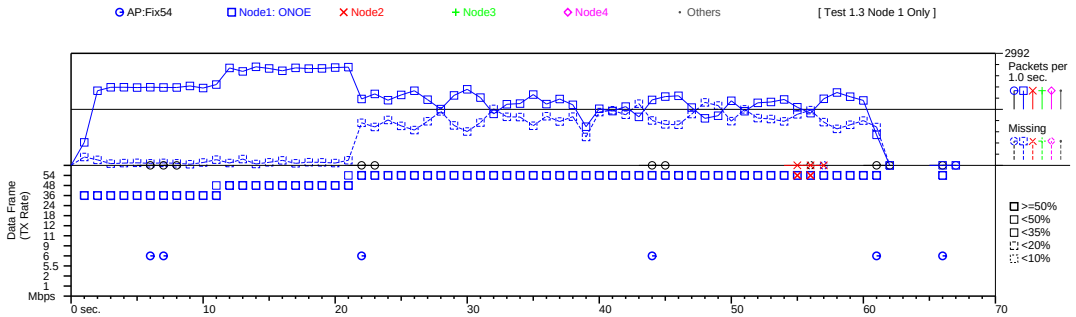
4.2.3 Test 1.3 - All clients use ONOE

No. of Active Node	Client 1	Client 2	Client 3	Client 4	Total
1	32.6				Individual
		32.6			Individual
			25.4		Individual
				29.8	Individual
2	16.4	16.7			33.1 Mbps
3	16.6	16.2	847K		33.6 Mbps
4	10.6	10.3	597K	6.9	28.4 Mbps

Table 4.3: Test 1.3 - ONOE - iperf server report

This is to test the performance of ONOE relative to the population. The result shows that the total goodput for single to three active nodes are around 33 Mbps which is close to the 35 Mbps reference channel bandwidth. The average aggregated utilization is relatively higher than ARF and AMRR. The performance drop is not serious as ARF case when number of active nodes increased. It is the most stable RCA in all test case.

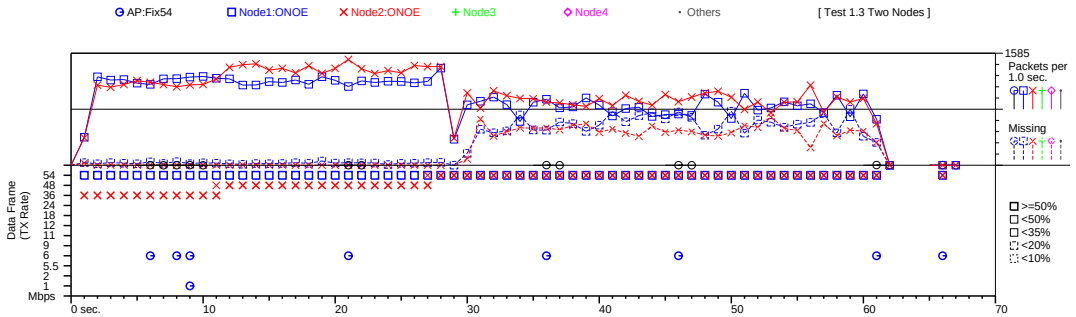
1. The single active node can use the whole channel to maximize the utilization. The ONOE uses 20 seconds to raise transmission rate to highest level. After that, no lower rate is used for data send.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 187152688 byte(s)
Total Org. Data:	6	112925	3	0	0	0	112934
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	1	2927	2	0	0	0	2930
Total ACK:	7	166471	5	0	0	0	166483
CRC Error:	0	0	0	0	0	0	0
Other Packet:	687	0	3	0	0	0	690
Missing Data:	1	51243	0	0	0		51244
Missing ACK:	1	37	0	0	0		38
Missing SEQ:	5	52437	25	0	0		52467

Figure 4.9: Test 1.3 - Single active client uses ONOE

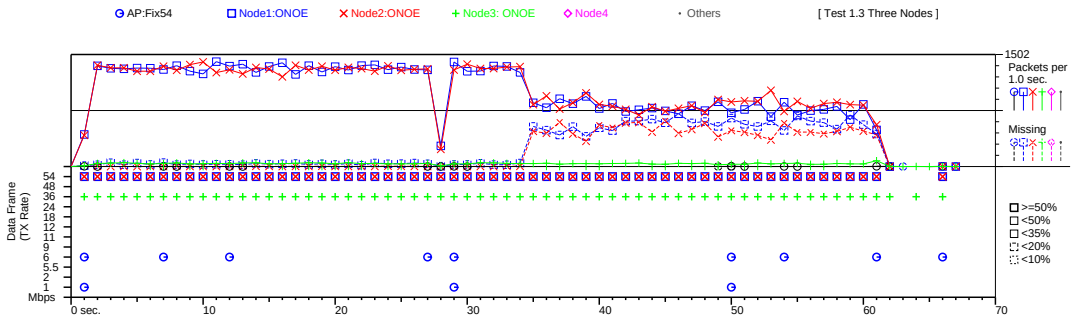
2. Similar to ARF and AMRR, the fairness on channel utilization is shown in two active node case. The channel utilization is still in high level. Most of transmission are still in highest rate range. The drop of packet count at the end of graph seems to be caused by the ability of the sniffer.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 214070336 byte(s)
Total Org. Data:	13	59616	64749	0	0	0	124378
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	2	3929	4814	0	0	0	8745
Total ACK:	10	82474	84313	0	0	0	166797
CRC Error:	0	0	0	0	0	0	0
Other Packet:	672	0	0	0	0	0	672
Missing Data:	0	19192	15453	0	0		34645
Missing ACK:	3	26	33	0	0		62
Missing SEQ:	12	20959	17120	0	0		38091

Figure 4.10: Test 1.3 - Two active clients use ONOE

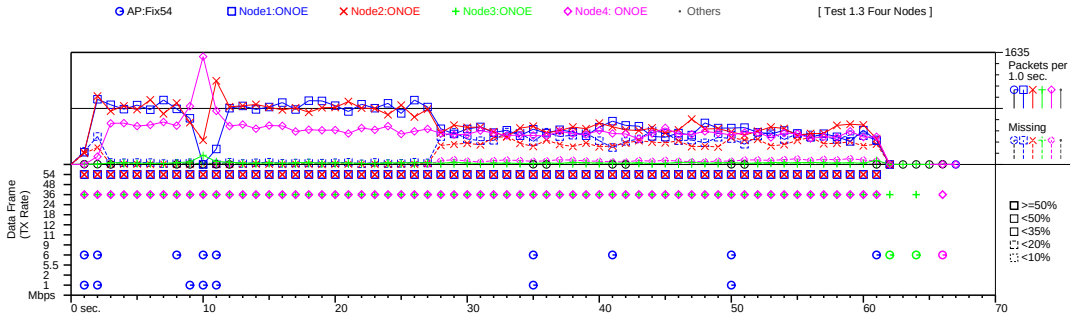
3. The fairness on channel utilization is shown in three active node case. The total goodput is still kept in high level. In contentious environment, no one can use highest transmission rate. But, there is one node to be over performance than others.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 227990584 byte(s)
Total Org. Data:	19	63947	64878	2436	0	0	131280
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	6	4333	4371	1975	0	0	10685
Total ACK:	17	84309	81950	4337	0	0	170613
CRC Error:	0	0	0	0	0	0	0
Other Packet:	677	0	0	6	0	0	683
Missing Data:	3	16314	13174	15	0		29506
Missing ACK:	7	27	27	0	0		61
Missing SEQ:	11	17621	14671	108	0		32411

Figure 4.11: Test 1.3 - Three active clients use ONOE

4. The fairness on channel utilization is still shown in four active node case. Even the total goodput is dropped. High rate transmission can be used. Much lower rate are used on retransmission. This slow down overall performance.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 199342586 byte(s)
Total Org. Data:	29	37032	38826	1512	30169	0	107568
Total Retx. Data:	0	0	1	0	0	0	1
Total Retransmit:	25	3813	4017	1288	7486	0	16629
Total ACK:	19	55286	53769	3144	35183	0	147401
CRC Error:	0	0	0	0	0	0	0
Other Packet:	682	0	0	0	0	0	682
Missing Data:	0	14583	11193	391	1939		28106
Missing ACK:	12	5	5	0	5		27
Missing SEQ:	4	14248	10899	299	1955		27405

Figure 4.12: Test 1.3 - Four active clients use ONOE

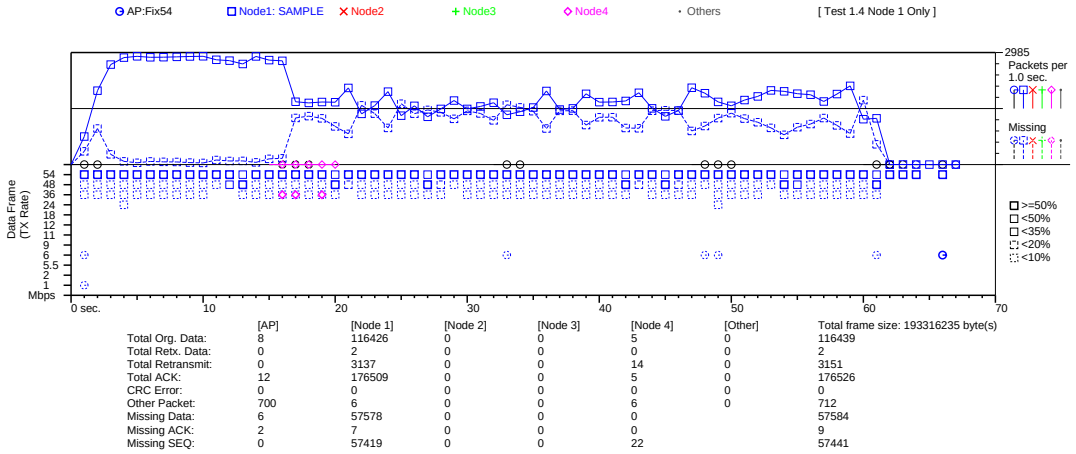
4.2.4 Test 1.4 - All clients use SAMPLE

No. of Active Node	Client 1	Client 2	Client 3	Client 4	Total
1	34.4				Individual
		34.2			Individual
			26.4		Individual
				31.1	Individual
2	16.7	16.4			33.1 Mbps
3	15.2	14.5	823K		30.5 Mbps
4	7.4	6.6	445K	3.9	18.3 Mbps

Table 4.4: Test 1.4 - SAMPLE - iperf server report

This is to test the performance of SAMPLE relative to the population. The result shows that the total goodput for single and two active nodes are around 33 Mbps which is close to the 35 Mbps reference channel bandwidth. The aggregated utilization is very similar to ONOE in few number of active nodes. The over all performance is still in higher then ARF and AMRR when number of active nodes increased.

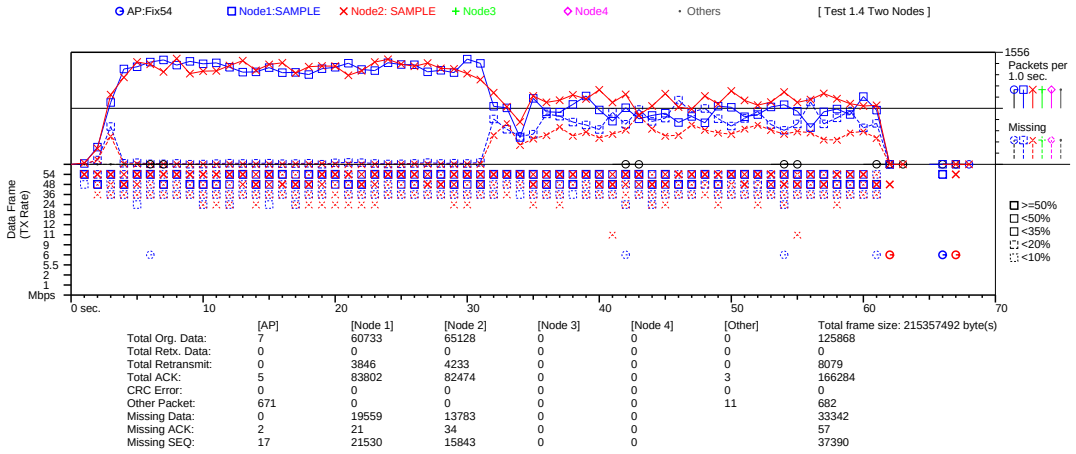
1. The single active node can use the whole channel to maximize the utilization. The SAMPLE will keep trying on different rate even there is only one active node.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 193316235 byte(s)
Total Org. Data:	8	116426	0	0	5	0	116439
Total Retx. Data:	0	2	0	0	0	0	2
Total Retransmit:	0	3137	0	0	14	0	3151
Total ACK:	12	176509	0	0	5	0	176526
CRC Error:	0	0	0	0	0	0	0
Other Packet:	700	6	0	0	6	0	712
Missing Data:	6	57578	0	0	0		57584
Missing ACK:	2	7	0	0	0		9
Missing SEQ:	0	57419	0	0	22		57441

Figure 4.13: Test 1.4 - Single active clients uses SAMPLE

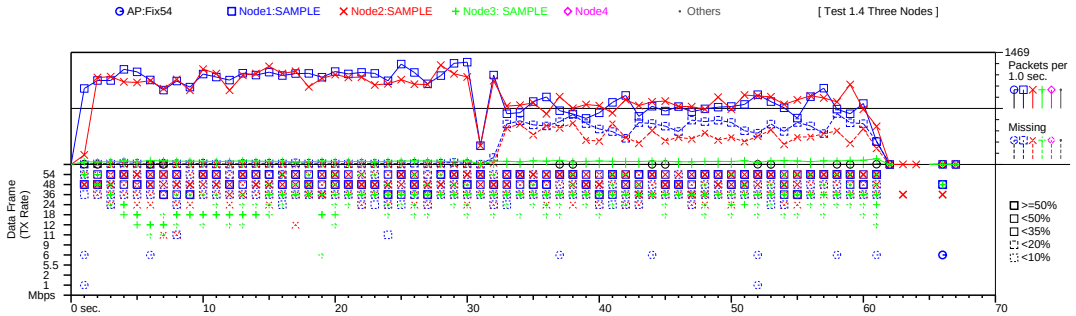
2. Same as above three RCA, the fairness on channel utilization is shown in two active node case. The channel utilization is still in high level. Most of transmission are still in high rate range. The range of rate trying is increased. The drop of packet count at the end of graph seems to be caused by the ability of the sniffer.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 215357492 byte(s)
Total Org. Data:	7	60733	65128	0	0	0	125868
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	0	3846	4233	0	0	0	8079
Total ACK:	5	83802	82474	0	0	3	166284
CRC Error:	0	0	0	0	0	0	0
Other Packet:	671	0	0	0	0	11	682
Missing Data:	0	19559	13783	0	0		33342
Missing ACK:	2	21	34	0	0		57
Missing SEQ:	17	21530	15843	0	0		37390

Figure 4.14: Test 1.4 - Two active clients use SAMPLE

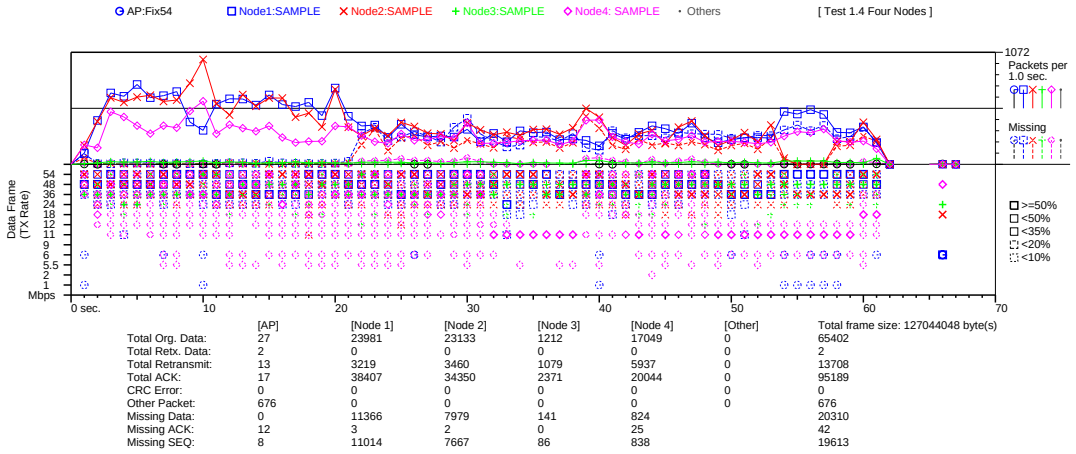
3. The fairness on channel utilization is shown in three active node case. The total goodput is still kept in high level. The behavior is no change for more nodes to be active.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 208072702 byte(s)
Total Org. Data:	15	57755	57322	2546	0	0	117638
Total Retx. Data:	0	2	0	0	0	0	2
Total Retransmit:	7	4980	5212	1732	0	0	11931
Total ACK:	13	77657	73254	4237	0	0	155161
CRC Error:	0	0	0	0	0	0	0
Other Packet:	686	0	2	0	0	0	688
Missing Data:	0	15216	11224	58	0		26498
Missing ACK:	4	13	21	4	0		42
Missing SEQ:	10	16308	12396	107	0		28821

Figure 4.15: Test 1.4 - Three active clients use SAMPLE

4. The individual frame rate jitter is large. There is no consistent frame rate to be maintained for streaming application. The transmission rate is changed frequently. This may be caused by large number of lower transmission rate is counted in statistic. It lets lower rate to be priority on transmission rate selection.



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 127044048 byte(s)
Total Org. Data:	27	23981	23133	1212	17049	0	65402
Total Retx. Data:	2	0	0	0	0	0	2
Total Retransmit:	13	3219	3460	1079	5937	0	13708
Total ACK:	17	38407	34350	2371	20044	0	95189
CRC Error:	0	0	0	0	0	0	0
Other Packet:	676	0	0	0	0	0	676
Missing Data:	0	11366	7979	141	824		20310
Missing ACK:	12	3	2	0	25		42
Missing SEQ:	8	11014	7667	86	838		19613

Figure 4.16: Test 1.4 - Four active clients use SAMPLE

4.2.5 Conclusion of RCA against population

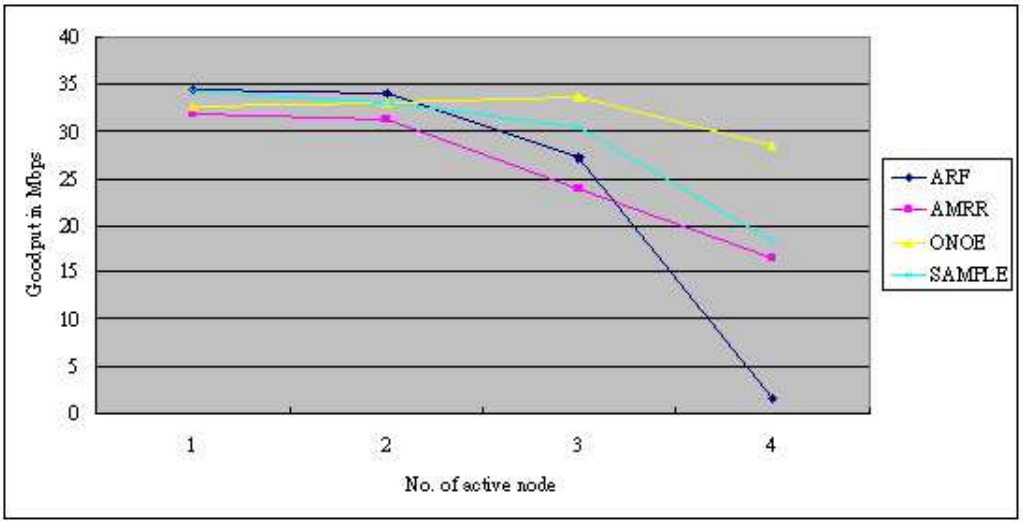


Figure 4.17: Goodput Summary of RCA against population

The Figure 4.17 is summary of test results in Table 4.1, 4.2, 4.3, and 4.4. It shows that the aggregated performance of ARF is very poor when number of active nodes increase. The AMRR, ONOE and SAMPLE have similar performance on small number of active nodes. The performance drop is not so high on population increasing and all of them are using same RCA. The ONOE is out performed on channel utilization.

4.3 Type 2 - Two RCA Competition

These tests are used to observes the effect and their efficacy when one node is using different RCA against others. I choose two newer RCA to compete with two popular RCA. The ONOE and SAMPLE are newer and complex in logic design. The ARF and AMRR are popular, simplex and straight in logic design. The objective of these experiment is to find any advantage of different logic design.

4.3.1 Test 2.1 - ONOE against ARF

Client	1	2	3	4
RCA	ONOE	ARF	ARF	ARF
Packet Sent	7474	4048	710	2302
Goodhput	1.5 Mbps	0.8 Mbps	0.1 Mbps	0.4 Mbps

Table 4.5: Test 2.1 - ONOE vs ARF - iperf server report

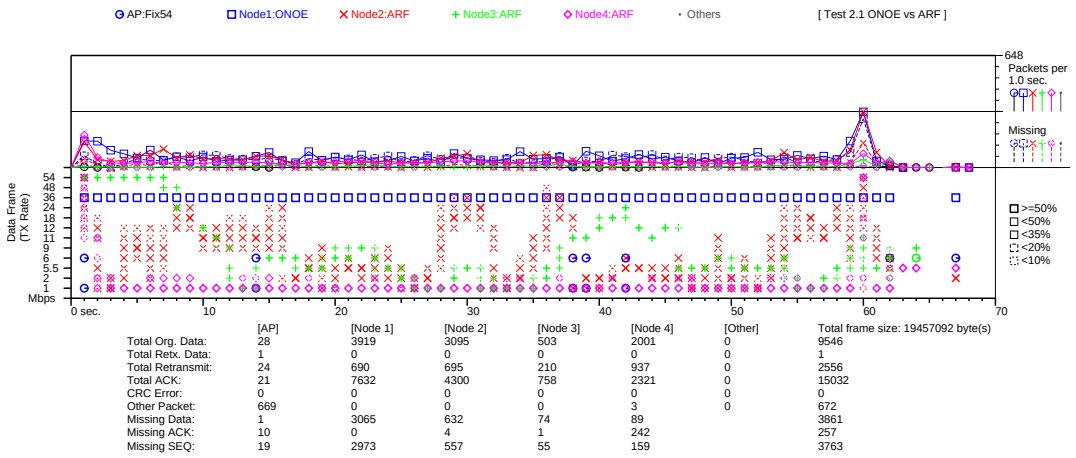


Figure 4.18: Test 2.1 - ONOE against ARF

As previous test for ARF in multi nodes contention, the aggregated goodput is very low if there are many ARF nodes. During contention, there is no one to raise up his transmission rate in higher level. ARF is poor in busy environment. It keeps using lowest rate for transmission and occupy a lot of air time. This slows down other nodes overall performance.

4.3.2 Test 2.2 - SAMPLE against ARF

Client	1	2	3	4
RCA	SAMPLE	ARF	ARF	ARF
Packet Sent	8177	4213	672	2173
Goodput	1.6 Mbps	0.8 Mbps	0.1 Mbps	0.4 Mbps

Table 4.6: Test 2.2 - SAMPLE vs ARF - iperf server report

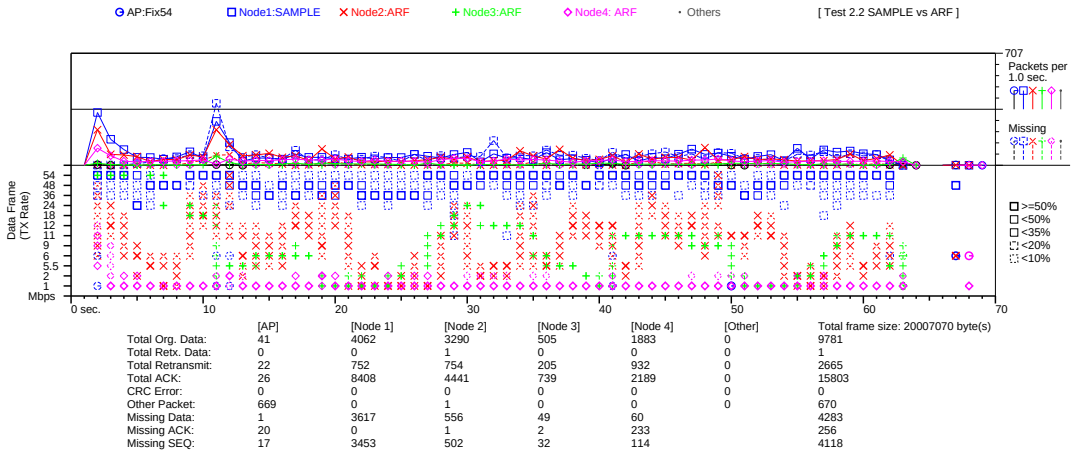


Figure 4.19: Test 2.2 - SAMPLE against ARF

Same as the case for ONOE against ARF, the aggregated goodput is very low. During contention, there is no ARF to raise up his transmission rate in higher level. ARF is poor in busy environment. SAMPLE is kept trying on different rate. But, multiple ARF nodes are using lowest rate to dominate a lot of air time. It makes the overall performance to be lower.

4.3.3 Test 2.3 - ONOE against AMRR

Client	1	2	3	4
RCA	ONOE	AMRR	AMRR	AMRR
Packet Sent	17946	18563	1875	10551
Goodput	3.5 Mbps	3.6 Mbps	0.4 Mbps	2.0 Mbps

Table 4.7: Test 2.3 - ONOE vs AMRR - iperf server report

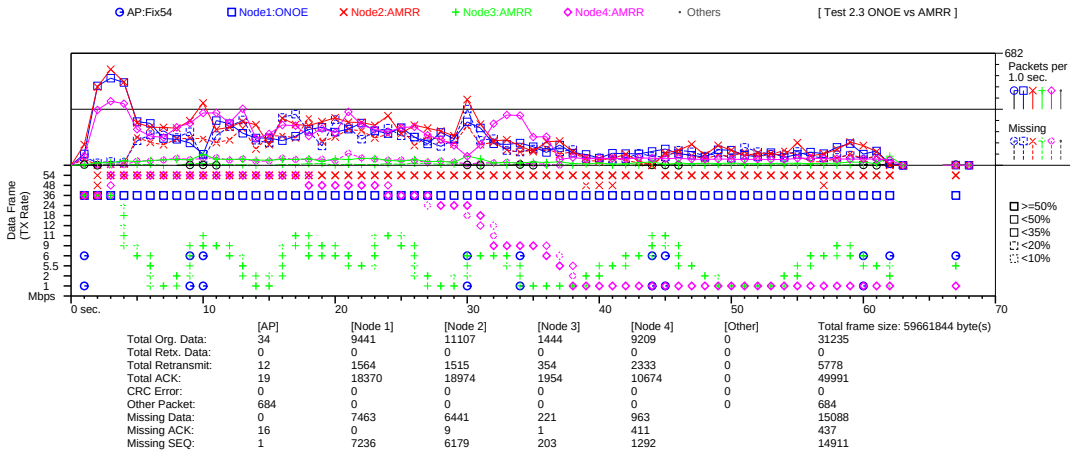


Figure 4.20: Test 2.3 - ONOE against AMRR

The aggregated goodput is higher than ARF comparison. The overall efficiency is not much different between ONOE and AMRR. Only one node drops his rate to lowest than more air time can be shared for others. Nodes using higher transmission rate can improve the goodput.

4.3.4 Test 2.4 - SAMPLE against AMRR

Client	1	2	3	4
RCA	Sample	AMRR	AMRR	AMRR
Packet Sent	20979	20257	2044	9239
Goodput	4.1 Mbps	4.0 Mbps	0.4 Mbps	1.8 Mbps

Table 4.8: Test 2.4 - SAMPLE vs AMRR - iperf server report

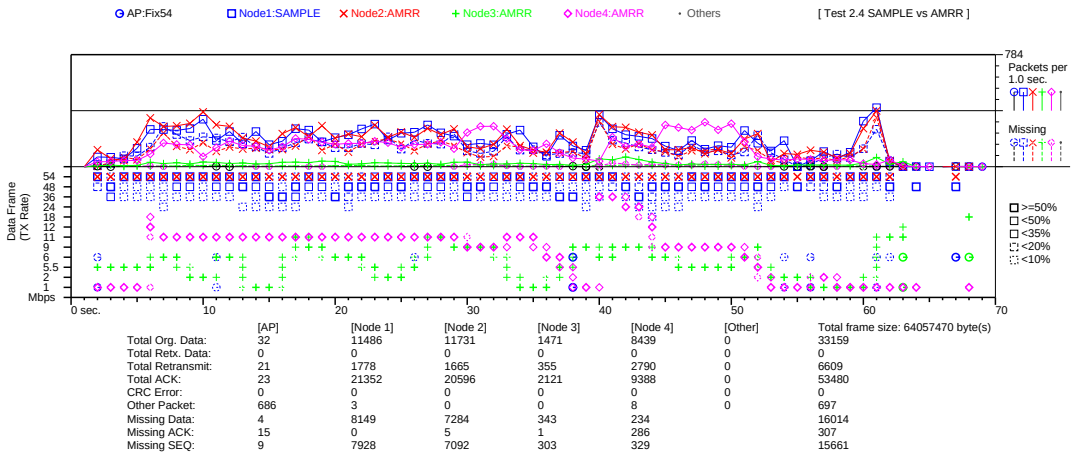


Figure 4.21: Test 2.4 - SAMPLE against AMRR

Same as the case for ONOE against AMRR, the aggregated goodput is higher than ARF comparison. Similar to ONOE case, only one node drops his rate to lowest than more air time can be shared for others. Who using higher transmission rate can improve overall goodput.

4.3.5 Summary for Two RCA Competition

There is no major advantage for any RCA against with others. Once again, multiple ARF active nodes let the network traffic jam which shown in Test 2.1 and 2.2. The Test 2.3 and 2.4 show well co-operation between AMRR and other RCA. Most of clients have similar share in bandwidth except the Node 3 that may be caused by its own problem.

4.4 Type 3 - Various RCA Competition

These tests are used to observes the effect and their efficacy when all active nodes are using different RCA. The ONOE and SAMPLE are represented the newer and complex in logic design. The ARF and AMRR are represented those popular, simplex and straight in logic design. The objective of these experiment is to find any advantage of different logic design.

4.4.1 Test 3.1 - ARF, AMRR, ONOE, SAMPLE

Client	1	2	3	4
RCA	ARF	AMRR	ONOE	SAMPLE
Packet Sent	30773	32636	2518	19115
Goodput	6.0 Mbps	6.4 Mbps	0.5 Mbps	3.7 Mbps

Table 4.9: Test 3.1 - ARF vs AMRR vs ONOE vs SAMPLE - iperf server report

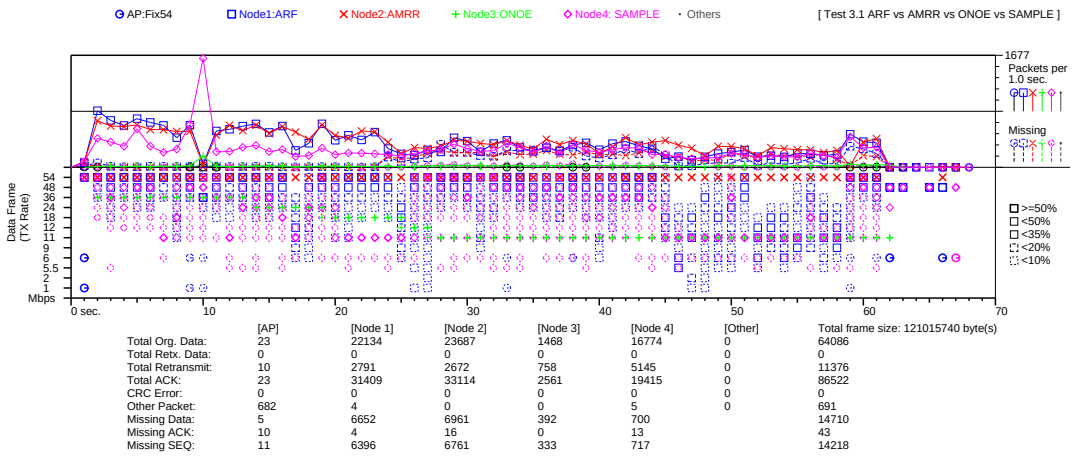


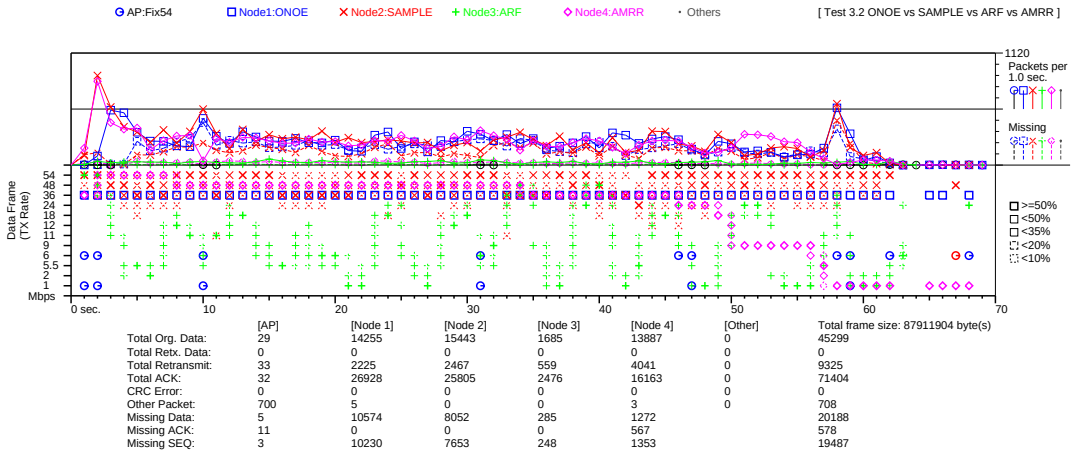
Figure 4.22: Test 3.1 - ARF, AMRR, ONOE, SAMPLE

The ARF is very sensitive on channel status, it switch to higher or lower rate frequently. The overall rate selection is in lower range that make its performance lower than AMRR and ONOE. In this test, the Client 4 cannot get use the channel, the lowest less throughput is the final effect.

4.4.2 Test 3.2 - ONOE, SAMPLE, ARF, AMRR

Client	1	2	3	4
RCA	ONOE	SAMPLE	ARF	AMRR
Packet Sent	26306	25131	2375	15972
Goodput	5.2 Mbps	4.9 Mbps	0.5 Mbps	3.1 Mbps

Table 4.10: Test 3.2 - ONOE vs SAMPLE vs ARF vs AMRR - iperf server report



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	
Total Org. Data:	29	14255	15443	1685	13887	0	45299
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	33	2225	2467	559	4041	0	9325
Total ACK:	32	26928	25805	2476	16163	0	71404
CRC Error:	0	0	0	0	0	0	0
Other Packet:	700	5	0	0	3	0	708
Missing Data:	5	10574	8052	285	1272		20188
Missing ACK:	11	0	0	0	567		578
Missing SEQ:	3	10230	7653	248	1353		19487

Figure 4.23: Test 3.2 - ONOE, SAMPLE, ARF, AMRR

It is similar to Test Case 3.1, the performance is hardware dependant.

4.4.3 Test 3.3 - SAMPLE, ONOE, AMRR, ARF

Client	1	2	3	4
RCA	SAMPLE	ONOE	AMRR	ARF
Packet Sent	10527	10310	982	3043
Goodput	2.1 Mbps	2.0 Mbps	0.2 Mbps	0.6 Mbps

Table 4.11: Test 3.3 - SAMPLE vs ONOE vs AMRR vs ARF - iperf server report

This shows that the performance is direct related to the transmission rate used. And, the lower RSSI causes lower channel competitive. The Node 4 use lower rate often and its RSSI is lower than others.

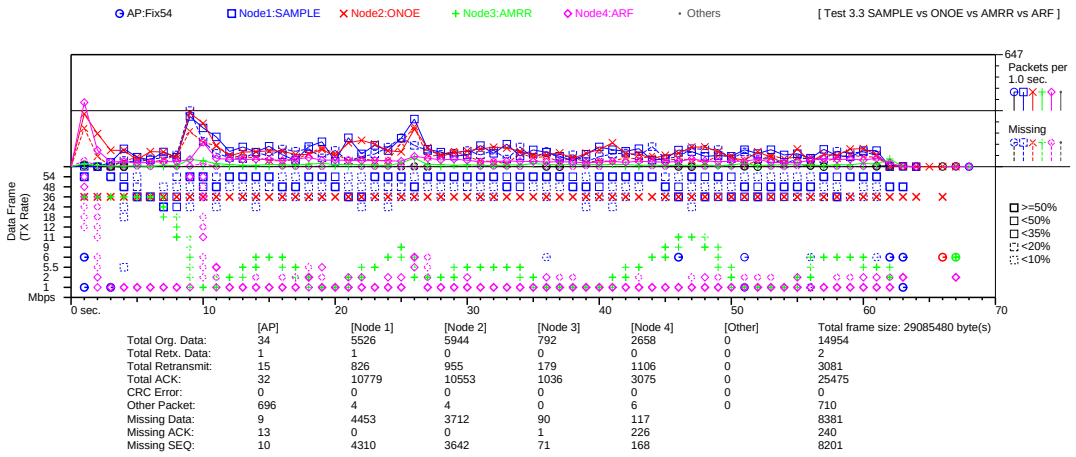


Figure 4.24: Test 3.3 - SAMPLE, ONOE, AMRR, ARF

4.4.4 Test 3.4 - AMRR, ARF, SAMPLE, ONOE

Client	1	2	3	4
RCA	AMRR	ARF	SAMPLE	ONOE
Packet Sent	40863	25527	2549	27385
Goodput	8.0 Mbps	5.0 Mbps	0.5 Mbps	5.4 Mbps

Table 4.12: Test 3.4 - AMRR vs ARF vs SAMPLE vs ONOE - iperf server report

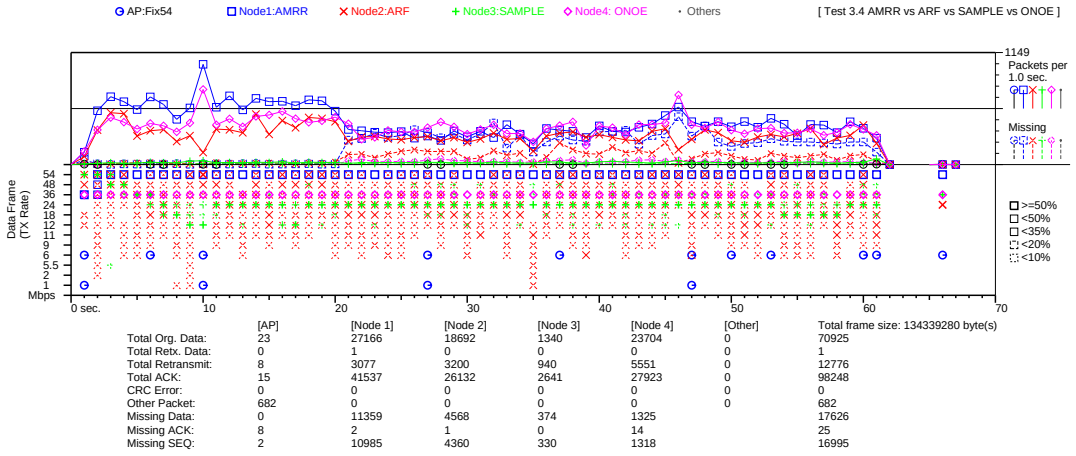


Figure 4.25: Test 3.4 - AMRR, ARF, SAMPLE, ONOE

Similar result as last Test Case 3.1 and 3.2.

4.4.5 Summary for Various RCA Competition

There is no major advantage for any RCA against with others. The performance of ARF is unstable, sometime is highest, sometime is lowest. On the other hand, AMRR, ONOE and SAMPLE is performed stable in most test. But, the total goodput is still low when number of active number is higher.

4.5 Recorded RCA Behavior

1. ARF — It is a sensitive algorithm. It always tries to raise up the transmission rate to high level or lower the rate once failure detected. That makes the efficiency of channel utilization very unstable in contentious network. In quiet environment, it provides high throughput as other algorithm.

2. AMRR — It is modified from ARF. It provides stable rate selection in all test. Overall performance is high and can cooperate with other algorithm in same network.

3. ONOE — It is the most stable rate selection algorithm in this study, not sensitive on channel status change. It requests longer time to confirm on raising of transmission rate to higher level. It is not so competitive on initial state. In challenging environment, it needs long time to push up the efficiency to higher level.

4. Sample — It is designed for adaptation. In above test, you will see that the throughput, number of data packet, is varying in larger range. It is due to frequently trying on different transmission rate. If there are not enough high rate count in statistical matrix, especially after long run, it seems difficult to select the high rate for transmission. It is not the sufficient case for streaming operation.

4.6 RCA Competition Summary

1. In terms of opportunity to obtain high throughput in a contentious network, the order of ability for those RCA is ARF > AMRR > Sample > ONOE. It is assumed that most of packet drops are caused by collision instead of frequent interference.

    - ARF — It is sensitive, always try to use high rate. And, its initial rate is highest available value and reducing when necessary.
    - AMRR — Less sensitive than ARF, raising threshold will be extended when transmission fail count increases. Its initial rate is lower value, i.e. 36 Mbps for 11g or 11 Mbps for 11b.
    - ONOE — Use long time to confirm the rate change situation. Its initial rate is same as AMRR, i.e. 36/11 Mbps for 11g/b.
    - Sample — Actively to try on different rate and select rate according to the statistical score, counting on successful and fail transmission. The jitter of rate change is large. The initial rate is also be 36 Mbps for 11g or 11 Mbps for 11b.

2. In terms of fairness, all RCA provides equal chance to use the channel if all nodes in a network are using same algorithm. No one will get obvious advantage by using higher rate than others. On the other hand, in test group 2 and 3, different RCA affects other algorithm performance. The overall behavior is depended on the population of RCA in the network. Figure 4.18 and 4.19 for ONOE and SAMPLE vs. ARF show the similar result as all ARF in Figure 4.4. It is same behavior for AMRR.

Chapter 5

Test on Auto-Selecting the Rate Adaptation Algorithm

5.1 The new concept RCA design

In tradition, there is only one RCA logic in each wireless NIC driver implementation. No matter of any network traffic character, all of packet transmission rate are controlled by the Rate Control Algorithm which is upper layer independent.

The design concept of the "Auto-Selecting the Rate Adaptation Algorithm" (ASRAA) is to develop a RCA to be adapt to application and channel status. The ASRAA consists of multiple RCA logic with policy based selection function. It can detect the sending application and channel status / statistic then check with defined policy to select appropriate RCA logic to handle transmission rate selection.

5.2 Overview of RCA logic in MadWiFi

For each packet to be sent, RCA functions will be call in sequence as Figure 5.1. There is only one RCA which is compiled with original MadWiFi design. The overview of interface is shown as follow.

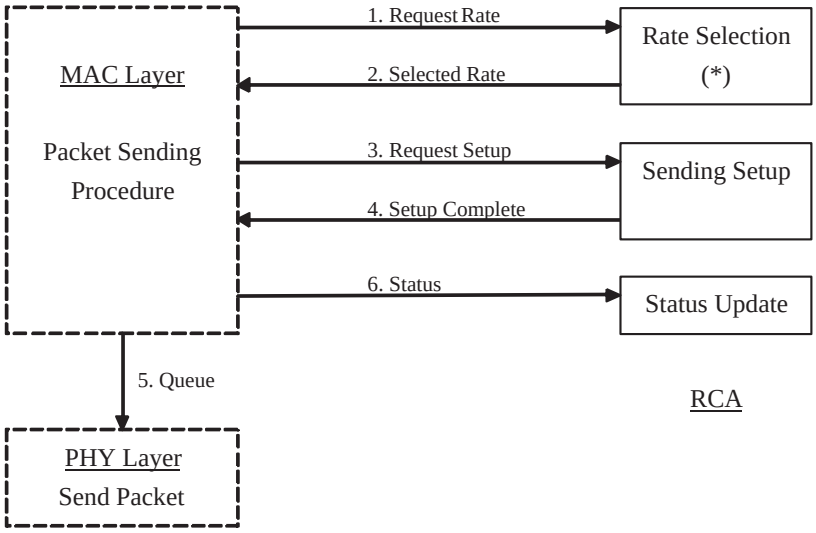


Figure 5.1: RCA calling sequence

(*) In original MadWiFi, only virtual NIC information and the frame size are passed into Rate Selection logic for rate selection. For ASRAA, the frame details will also be passed into Rate Selection logic for packet analysis. With detected type of data and application, an appropriate RCA will be elected to process the rate selection.

5.3 The ASRAA

5.3.1 Auto-RCA-Select Function

Unlike other traditional RCA implementation, the sending packet's details will be passing into the function of "Auto-RCA-Select". After analysis on packet header, the calling application will be detected. With the knowledge of application and according to the predefined policy, the appropriate RCA will be used to process the rate selection.

5.3.2 Application Detection

To detect the application, the most direct way is to analysis the packet's header which obtains calling application information. In the header, we can find protocol, ports, frame type and etc. For example,

1. Use LLC & SNAP to detect frame type.
2. Use IP Header to detect protocol, such as TCP, UDP, ICMP, etc.
3. Use IP Address to identify destination.
4. Use Port Number to determinate current application, such as 25 for SMTP, 80 for HTTP, 21 for FTP.

5.4 Test on ASRAA

Application Running	cat /proc/asraa_conf
FTP	ARF
VLC Video Streaming	AMRR
Any others	Sample

Table 5.1: ASRAA Auto-RCA-Select Sample Policy

As the predefined policy table, the specified RCA is selected according on the application run. This proved that the framework of Auto-RCA-Select is work, further development can be carried on.

Chapter 6

Conclusion

6.1 New tools are helpful

1. XGraph — It provides rich information for wireless network traffic analysis. Multiple views of result are time aligned for easy to trace.

2. RemoteCall and RemoteExecute — This platform independent remote control system is very helpful for multi nodes network measurement control. It helps me to perform all tests through the central controller to control various node in the network.

6.2 Sniffer Choice

As the test result for Sniffing Methodology, the guide line to setup a sniffer is:

1. The processor speed should be at least 1.5GHz which can handle 54M wireless network sniffing.

2. MadWiFi is better than AirPcap on data sniffing.

3. Use AirPcap if you want to capture any packet like from the channel. The MadWiFi in monitor mode can capture packets with correct FCS value only.

4. If possible, use RAM Disk or Solid State Disk as the working storage to eliminate the normal disk write to occupy lot of system resource. It is because Disk write will slow down the capturing.

5. Keep in mind on Sequence Number Reorder in packet tracing / analysis.

6.3 RCA Performance

In the RCA performance comparison, the ARF is behind than others in congested networks. Moreover, it is not as cooperative as other algorithms. All other tested algorithms — AMRR, ONOE and SAMPLE — are working fine on various scenarios, no matter on contentious network or cooperative manner. Finally, the AMRR is an effective RCA in this study. It provides higher throughput in most cases. Rate change is not frequently. The ONOE is stable and relatively higher performance. It can be used for those applications which require the streaming function to be stable. SAMPLE changes its rate too often.

6.4 ASRAA

Finally, the framework of the Auto-RCA-Select for ASRAA is proved to work even the policy is not well defined. It bring out the new idea of rate control for wireless network, RCA is according to application running or environmental status.

Chapter 7

Future Work

As the conclusion on last chapter, there are some works to be done on future.

1. Extend the Auto-RCA-Select function in "ASRAA" to be smart enough on algorithm selection. Apply the collected information, such as RCA behavior and application requirement, into the switch logic. Produce more practical mapping between RCA and application.

2. Study the cause of excessive decreased on overall goodput which is observed during RCA comparison in high contention test. Is the effect of RCA control or 802.11 backoff algorithm ?

3. Test more capturing device to improve sniffing efficiency.

Bibliography

[1] Airpcap: Passive capturer for wifi. available from CACE Technologies. [cited at p. IV]

[2] iperf: Network performance measurement. available from http://dast.nlanr.net/projects/Iperf. [cited at p. 26, 31]

[3] Madwifi: Bit-rate selection algorithms. available from http://madwifi.org/wiki/UserDocs/RateControl. [cited at p. V, 1, 3, 41]

[4] Madwifi: Multiband atheros driver for wifi. available from http://www.madwifi.org. [cited at p. IV, 5]

[5] Rssi: Received signal strength indication. available from http://madwifi.org/wiki/UserDocs/RSSI. [cited at p. 5]

[6] J. Bicket. Bit-rate selection in wireless networks. Master's thesis, MIT, Feb 2005. [cited at p. V, 1, 3, 4, 41]

[7] A. Kamerman and L. Monteban. *WaveLAN-II: A high-performance wireless LAN for the unlicensed band.* Bell Labs Technical Journal, Summer 1997. [cited at p. V, 1, 2, 3, 41]

[8] M. Manshaei M. Lacage and T. Turletti. Ieee 802.11 rate adaptation: A practical approach. Proc. ACM MSWiM, Oct 2004. [cited at p. V, 1, 3, 41]

Appendices

Appendix A

Command Setup for RCA Performance Analysis

A.1 Server Initialization

1. Set IP to 192.168.10.10 for the wired NIC
2. Set wireless NIC to channel 2
3. Set IP to 192.168.100.254 for the wireless NIC
4. Run "echo fix 54 >/proc/asraa_conf" to set RCA to Fix 56
5. Run "iperf -s -p -u 2001" to wait for Client 1 data
6. Run "iperf -s -p -u 2002" to wait for Client 2 data
7. Run "iperf -s -p -u 2003" to wait for Client 3 data

A.2 Sniffer Initialization

1. Set IP to 192.168.10.11 for the wired NIC

2. Set wireless NIC to channel 2

3. Set IP to 192.168.100.11 for the wireless NIC named “ath0”

4. Set up MadWiFi Monitor VAP named “ath1”

5. Run “echo fix 54 >/proc/asraa_conf” to set RCA to Fix 56

A.3 Client Initialization

1. Set IP to 192.168.10.20x for the wired NIC, where x is client number

2. Set wireless NIC to channel 2

3. Set IP to 192.168.100.x for the wireless NIC, where x is client number

4. Run “RemoteCallExecute” to wait for commands from Controller

A.4 Controller Initialization

1. Set IP to 192.168.10.101 for the wired NIC

2. Run “ssh 192.168.10.201” to start remote monitor on Client 1

3. Run “ssh 192.168.10.202” to start remote monitor on Client 2

4. Run “ssh 192.168.10.203” to start remote monitor on Client 3

5. Run “RemoteCall” to start remote control

A.5 Remote Command Setup in Clients

1. Use “RemoteCall” to send “echo RCA >/proc/asraa_conf; iperf -c 192.168.102.254 -p 2001 -u -l 1470 -b 35M -t 60” to Client 1, the RCA setting is according to the test case.

2. Use “RemoteCall” to send “echo RCA >/proc/asraa_conf; iperf -c 192.168.102.254 -p 2002 -u -l 1470 -b 35M -t 60” to Client 2, the RCA setting is according to the test case.

3. Use “RemoteCall” to send “echo RCA >/proc/asraa_conf; iperf -c 192.168.102.254 -p 2003 -u -l 1470 -b 35M -t 60” to Client 3, the RCA setting is according to the test case.

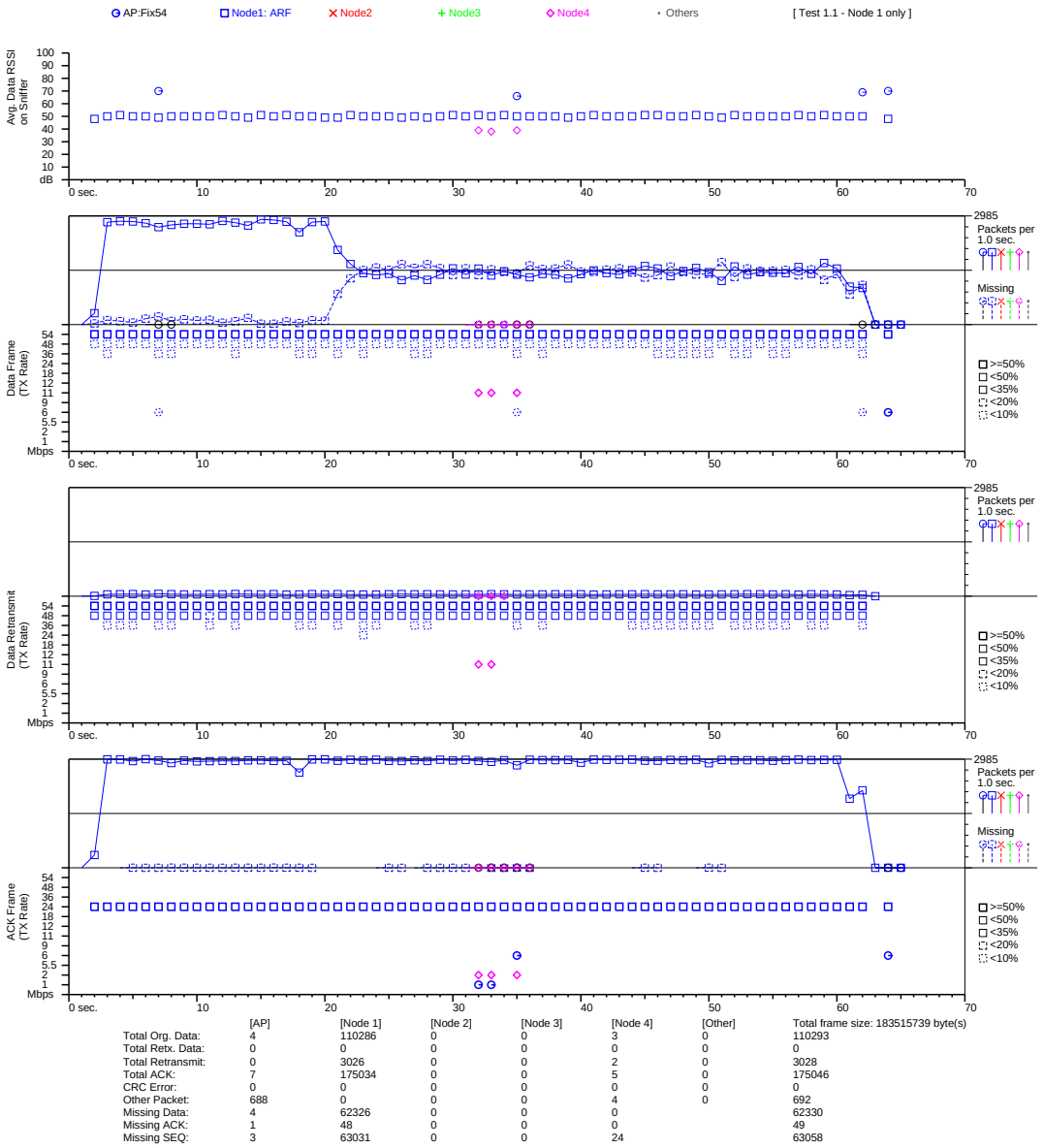
A.6 Trigger Execution in Sniffer

1. Run “tshark -i ath1 -a duration:70 -w test.cap” in Sniffer to start capture

2. In Controller, use “RemoteCall” to trigger the execution by click [Execute] button

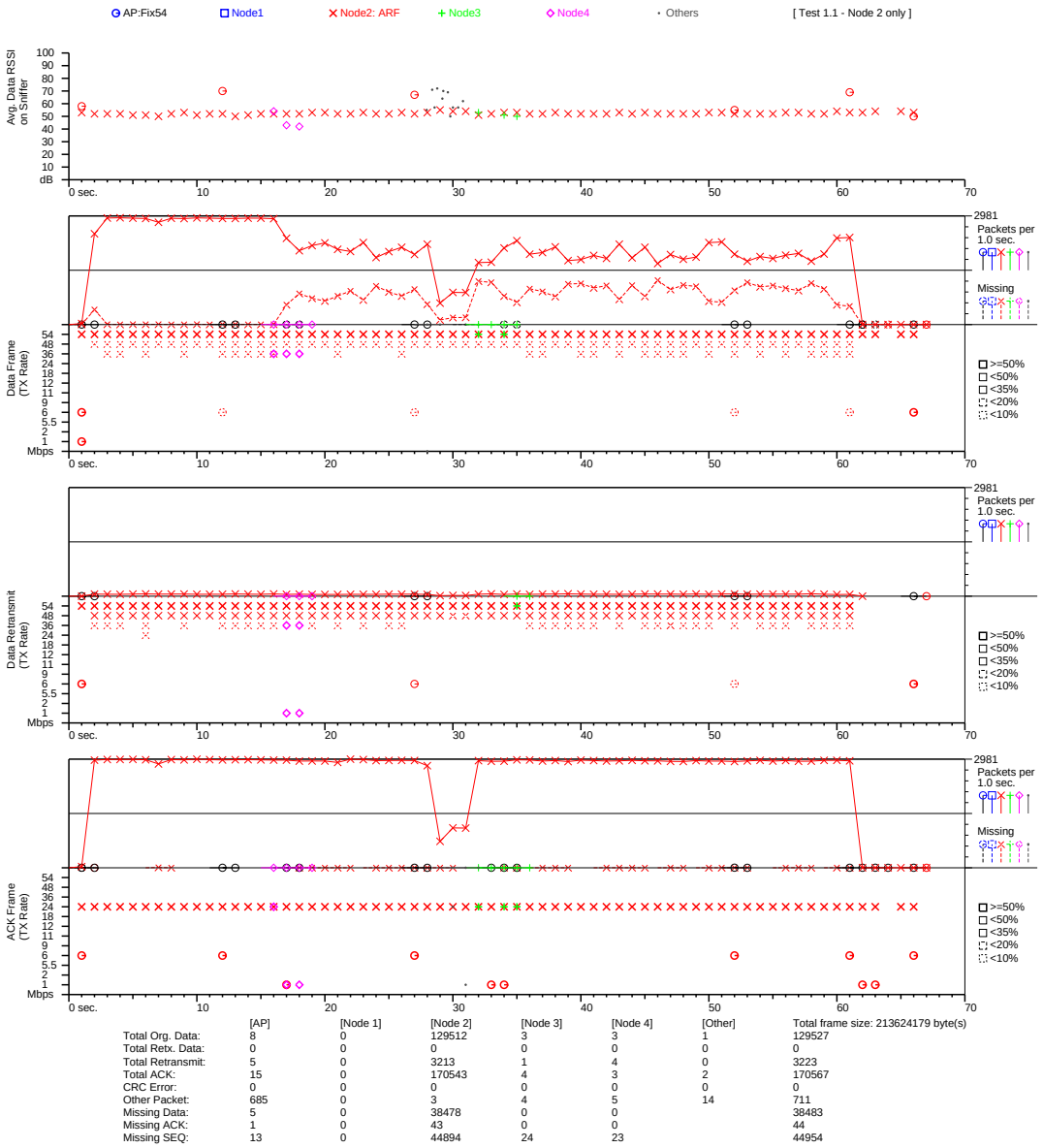
Appendix B

XGraph Output for RCA Performance Analysis



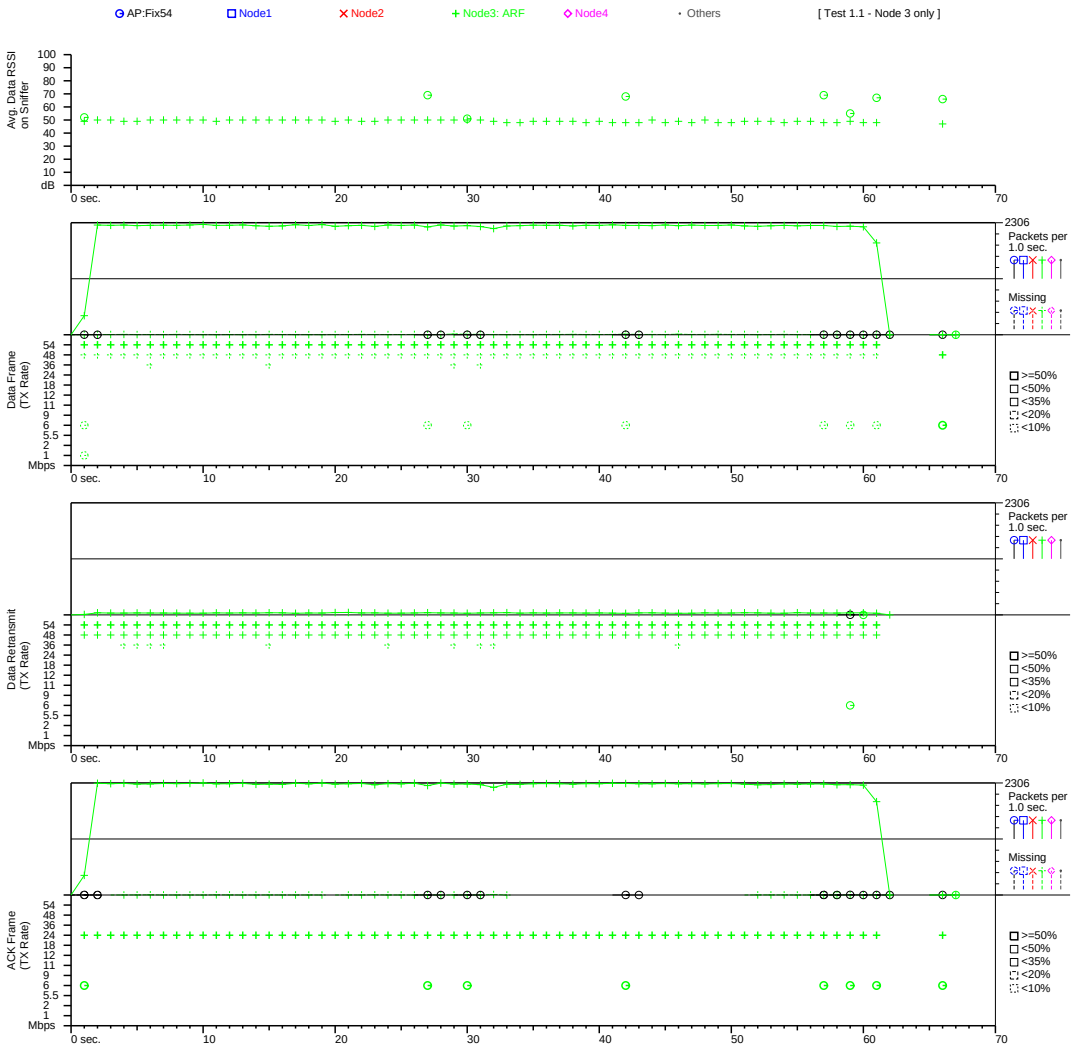
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 183515739 byte(s)
Total Org. Data:	4	110286	0	0	3	0	110293
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	0	3026	0	0	2	0	3028
Total ACK:	7	175034	0	0	5	0	175046
CRC Error:	0	0	0	0	0	0	0
Other Packet:	688	0	0	0	4	0	692
Missing Data:	4	62326	0	0	0		62330
Missing ACK:	1	48	0	0	0		49
Missing SEQ:	3	63031	0	0	24		63058

Figure B.1: Test 1.1 - Only Note 1 uses ARF (Full Graph)



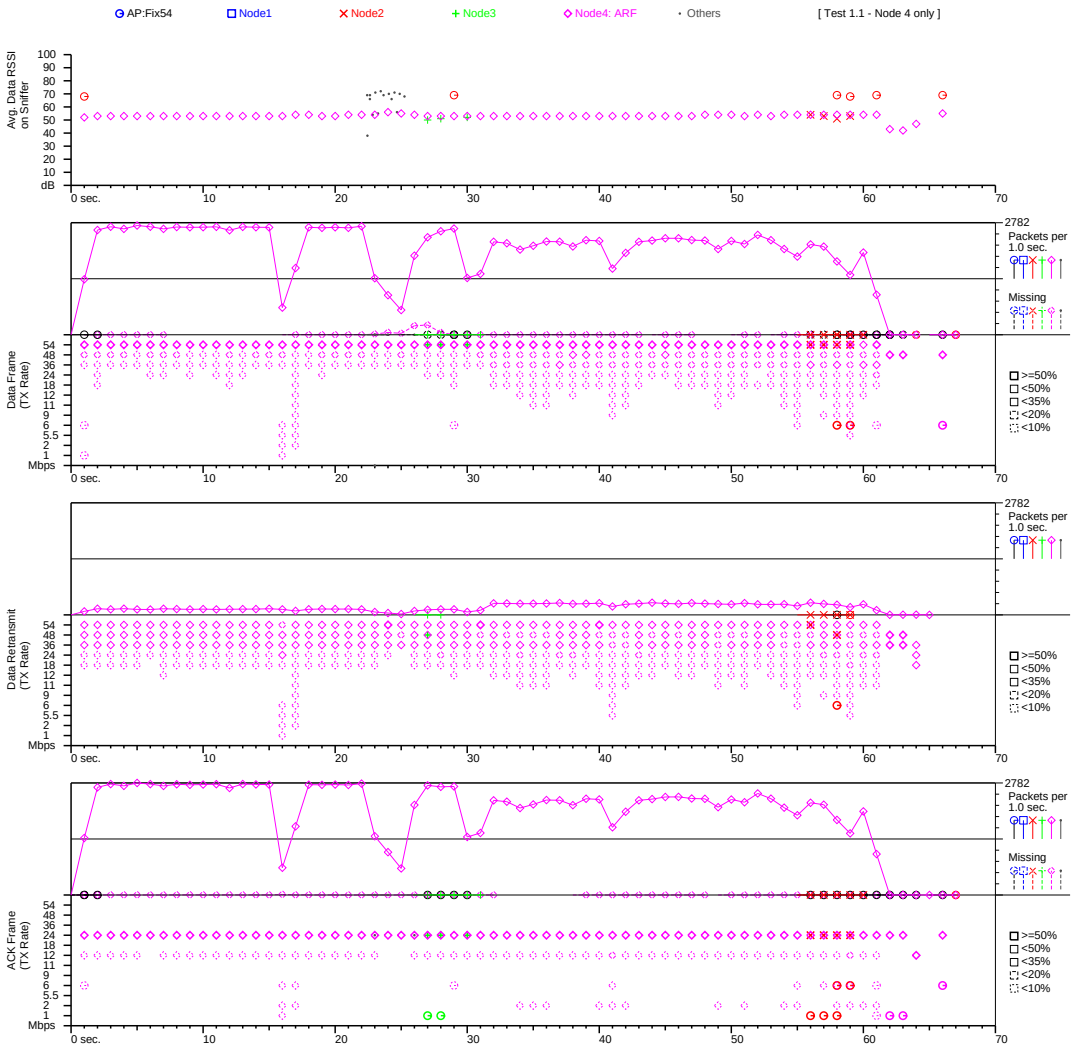
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 213624179 byte(s)
Total Org. Data:	8	0	129512	3	3	1	129527
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	5	0	3213	1	4	0	3223
Total ACK:	15	0	170543	4	3	2	170567
CRC Error:	0	0	0	0	0	0	0
Other Packet:	685	0	3	4	5	14	711
Missing Data:	5	0	38478	0	0		38483
Missing ACK:	1	0	43	0	0		44
Missing SEQ:	13	0	44894	24	23		44954

Figure B.2: Test 1.1 - Onle Node 2 uses ARF (Full Graph)



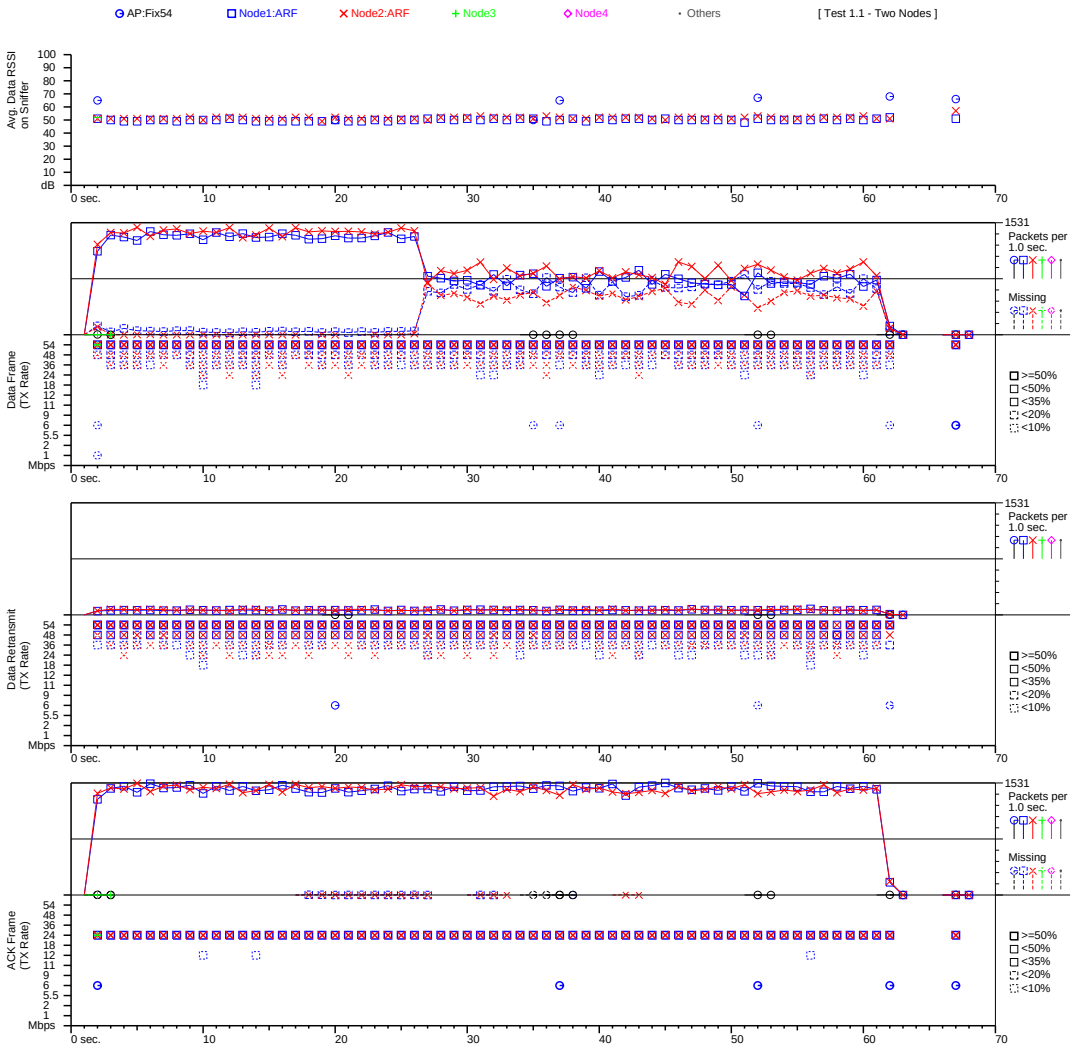
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 219341930 byte(s)
Total Org. Data:	10	0	0	134857	0	0	134867
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	2	0	0	2346	0	0	2348
Total ACK:	10	0	0	137145	0	0	137155
CRC Error:	0	0	0	0	0	0	0
Other Packet:	681	0	0	0	0	0	681
Missing Data:	0	0	0	330	0		330
Missing ACK:	2	0	0	63	0		65
Missing SEQ:	3	0	0	587	0		590

Figure B.3: Test 1.1 - Only Note 3 uses ARF (Full Graph)



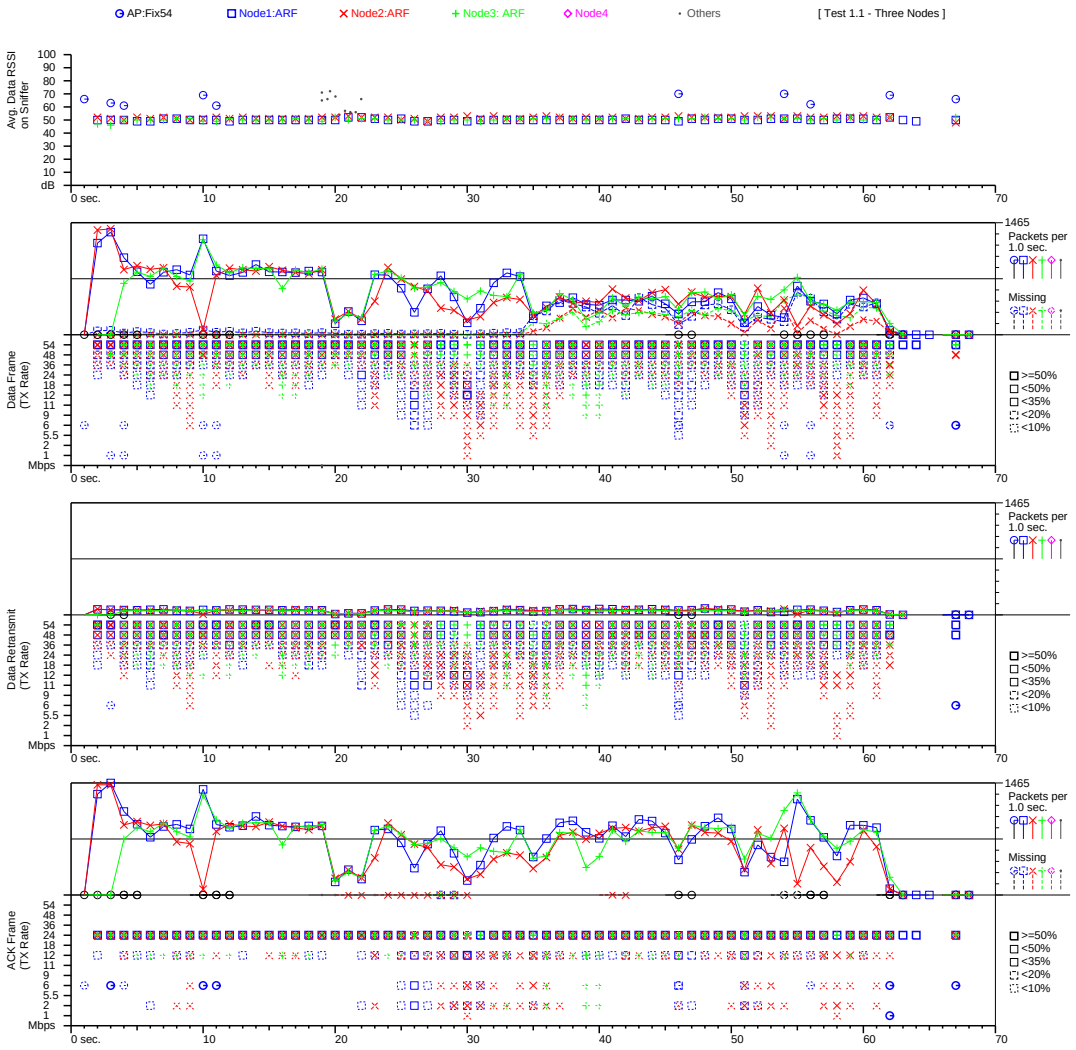
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 233421490 byte(s)
Total Org. Data:	7	0	4	4	134387	1	134403
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	1	0	2	1	11831	0	11835
Total ACK:	14	0	7	5	138125	3	138154
CRC Error:	0	0	0	0	0	0	0
Other Packet:	683	0	5	3	4	14	709
Missing Data:	8	0	1	0	665		674
Missing ACK:	1	0	0	0	86		87
Missing SEQ:	29	0	24	25	9446		9524

Figure B.4: Test 1.1 - Only Note 4 uses ARF (Full Graph)



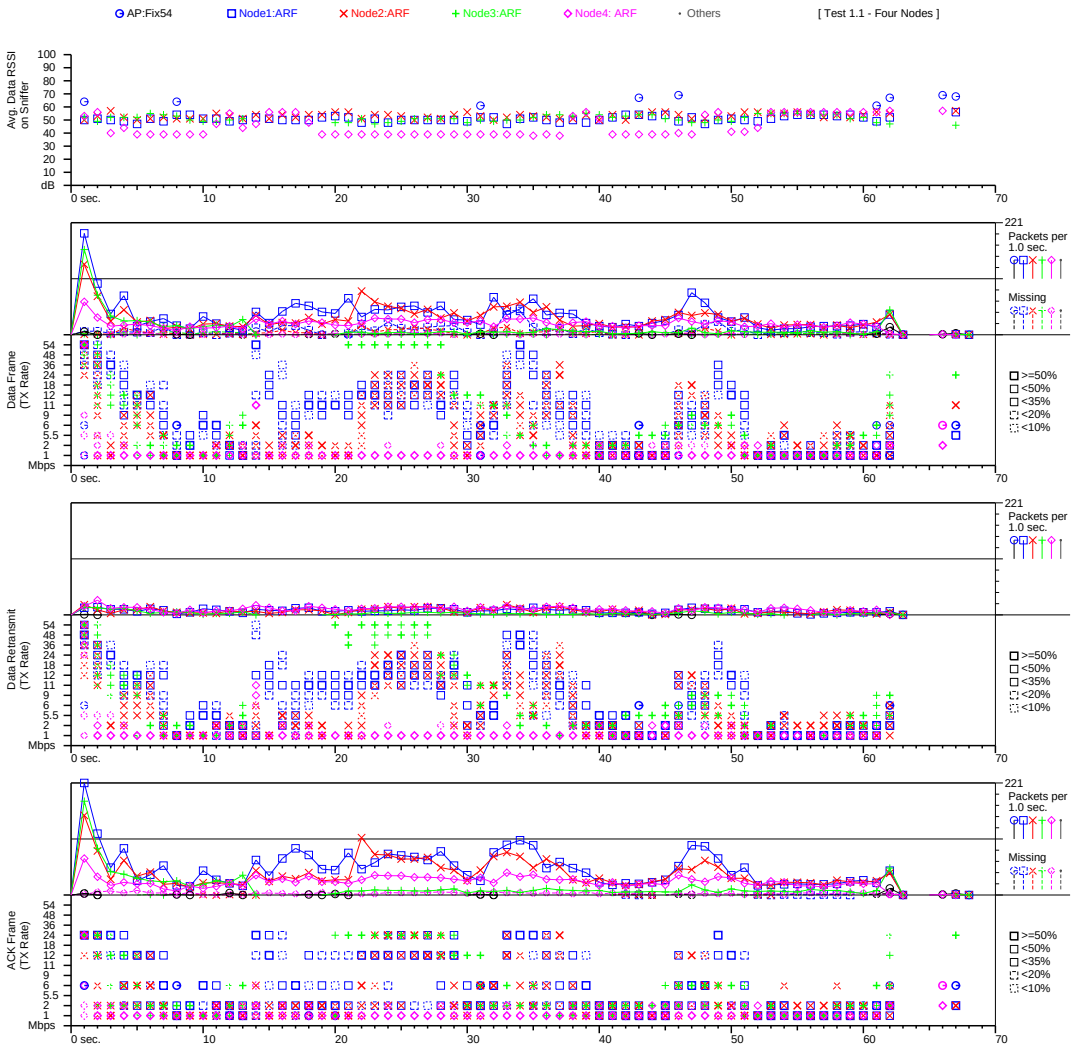
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 213157782 byte(s)
Total Org. Data:	11	59644	64750	1	0	0	124406
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	3	3826	4106	0	0	0	7935
Total ACK:	8	87435	87126	1	0	0	174570
CRC Error:	0	0	0	0	0	0	0
Other Packet:	684	0	0	0	0	0	684
Missing Data:	0	24236	18859	0	0		43095
Missing ACK:	4	14	18	0	0		36
Missing SEQ:	1	24284	19032	0	0		43317

Figure B.5: Test 1.1 - Two active clients use ARF (Full Graph)



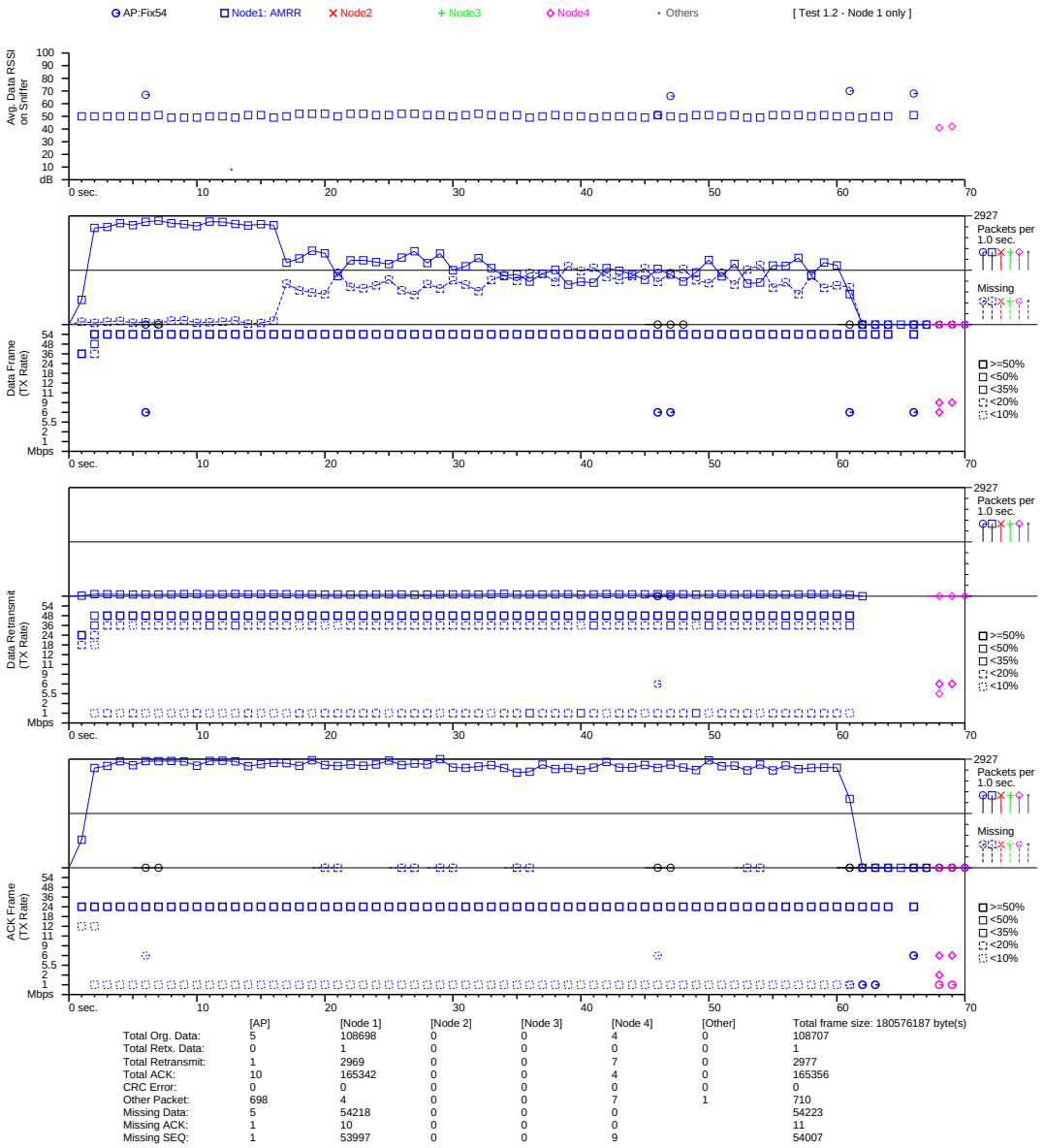
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 178590479 byte(s)
Total Org. Data:	19	34745	32833	33632	0	1	101230
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	6	3491	3444	3040	0	0	9981
Total ACK:	12	47719	41084	44456	0	2	133273
CRC Error:	0	0	0	0	0	0	0
Other Packet:	679	4	0	0	0	13	696
Missing Data:	1	9680	5249	7927	0		22857
Missing ACK:	8	1	8	2	0		19
Missing SEQ:	11	11510	7197	9753	0		28471

Figure B.6: Test 1.1 - Three active clients use ARF (Full Graph)



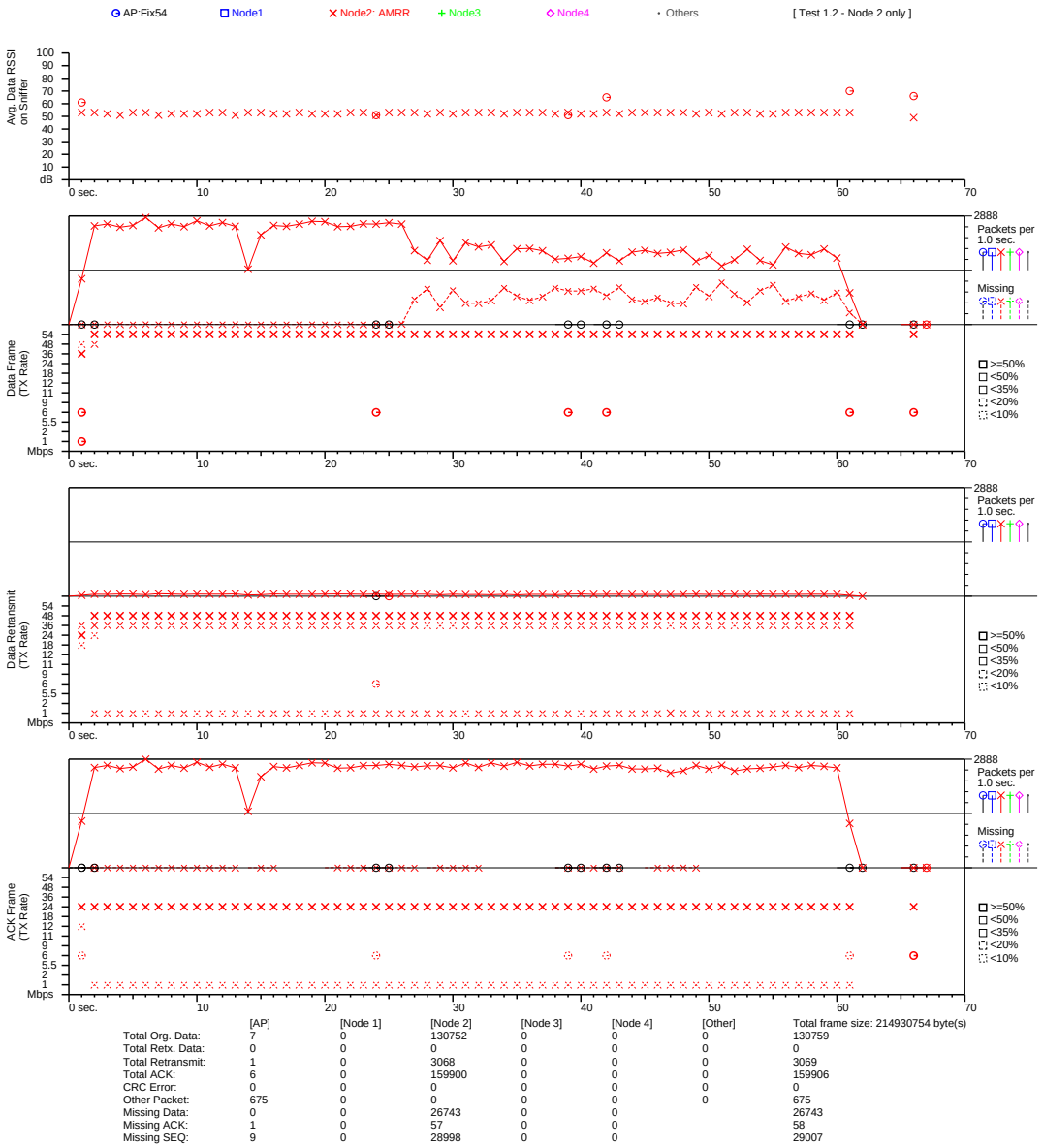
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 13731360 byte(s)
Total Org. Data:	35	2396	2078	748	1372	0	6629
Total Retx. Data:	1	0	0	1	0	0	2
Total Retransmit:	9	516	513	207	701	0	1946
Total ACK:	29	3371	2722	1004	1670	0	8796
CRC Error:	0	0	0	0	0	0	0
Other Packet:	671	1	1	7	1	0	681
Missing Data:	7	525	255	75	60		922
Missing ACK:	13	6	8	2	199		228
Missing SEQ:	27	470	218	72	131		918

Figure B.7: Test 1.1 - Four active clients use ARF (Full Graph)



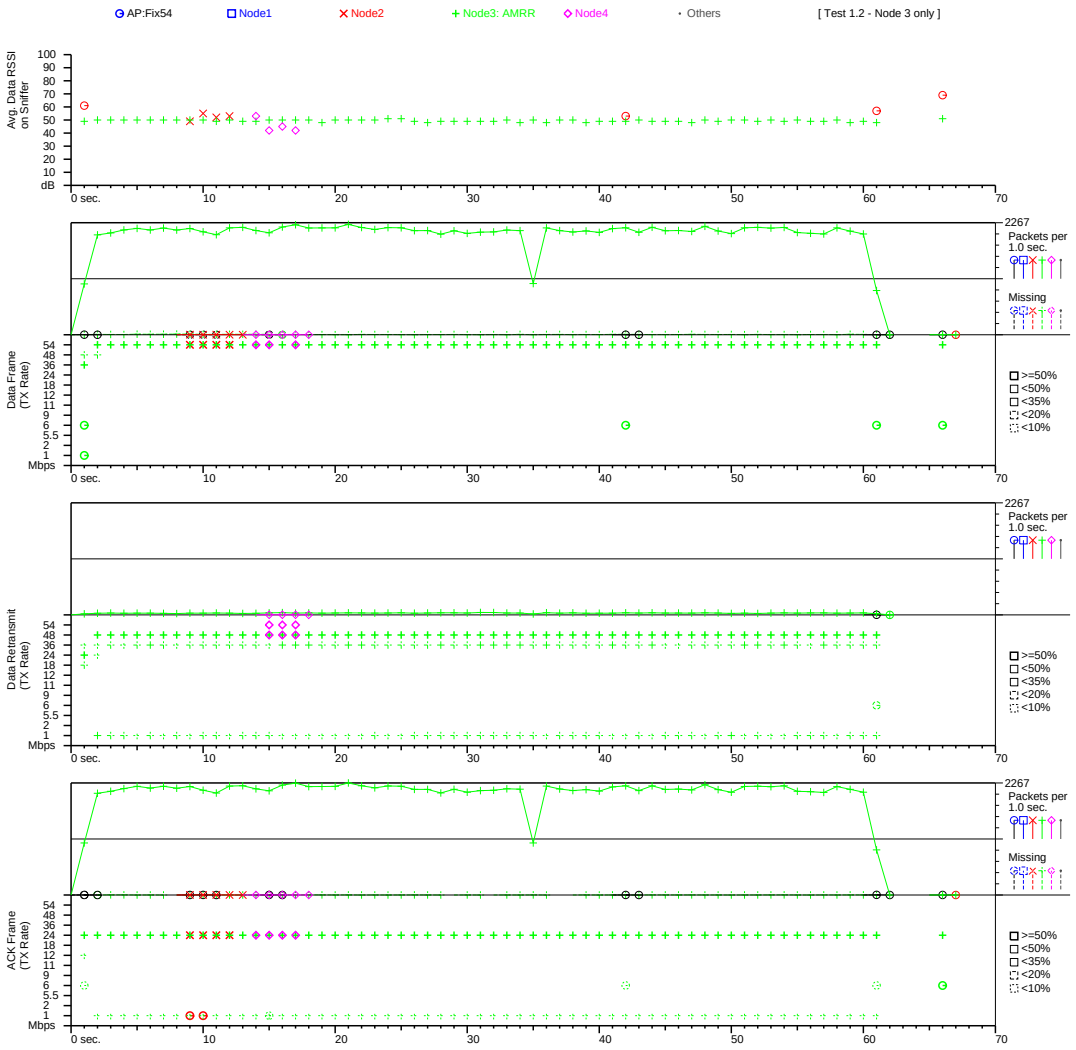
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	
Total Org. Data:	5	108698	0	0	4	0	108707
Total Retx. Data:	0	1	0	0	0	0	1
Total Retransmit:	1	2969	0	0	7	0	2977
Total ACK:	10	165342	0	0	4	0	165356
CRC Error:	0	0	0	0	0	0	0
Other Packet:	698	4	0	0	7	1	710
Missing Data:	5	54218	0	0	0		54223
Missing ACK:	1	10	0	0	0		11
Missing SEQ:	1	53997	0	0	9		54007

Figure B.8: Test 1.2 - Only Node 1 uses AMRR (Full Graph)



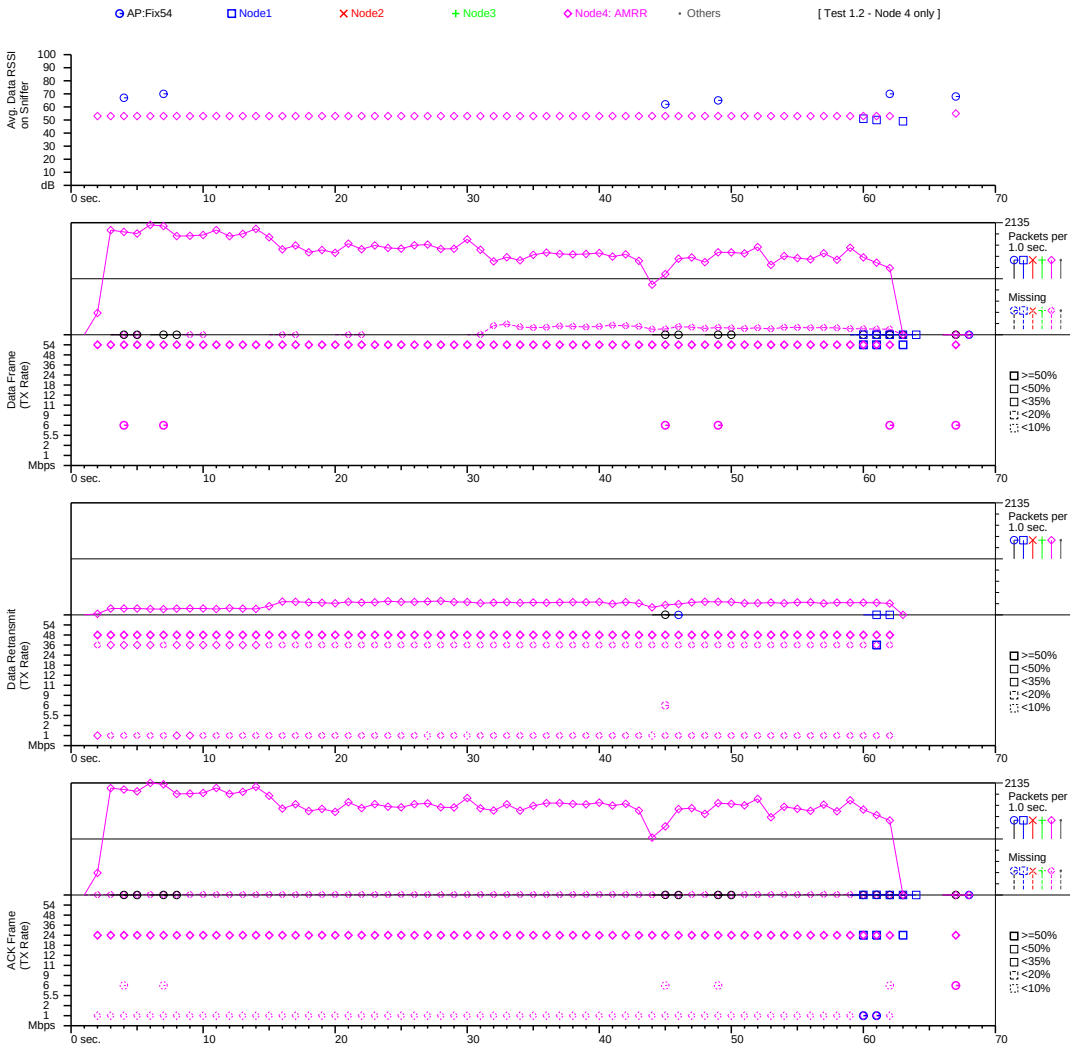
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 214930754 byte(s)
Total Org. Data:	7	0	130752	0	0	0	130759
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	1	0	3068	0	0	0	3069
Total ACK:	6	0	159900	0	0	0	159906
CRC Error:	0	0	0	0	0	0	0
Other Packet:	675	0	0	0	0	0	675
Missing Data:	0	0	26743	0	0		26743
Missing ACK:	1	0	57	0	0		58
Missing SEQ:	9	0	28998	0	0		29007

Figure B.9: Test 1.2 - Only Node 2 uses AMRR (Full Graph)



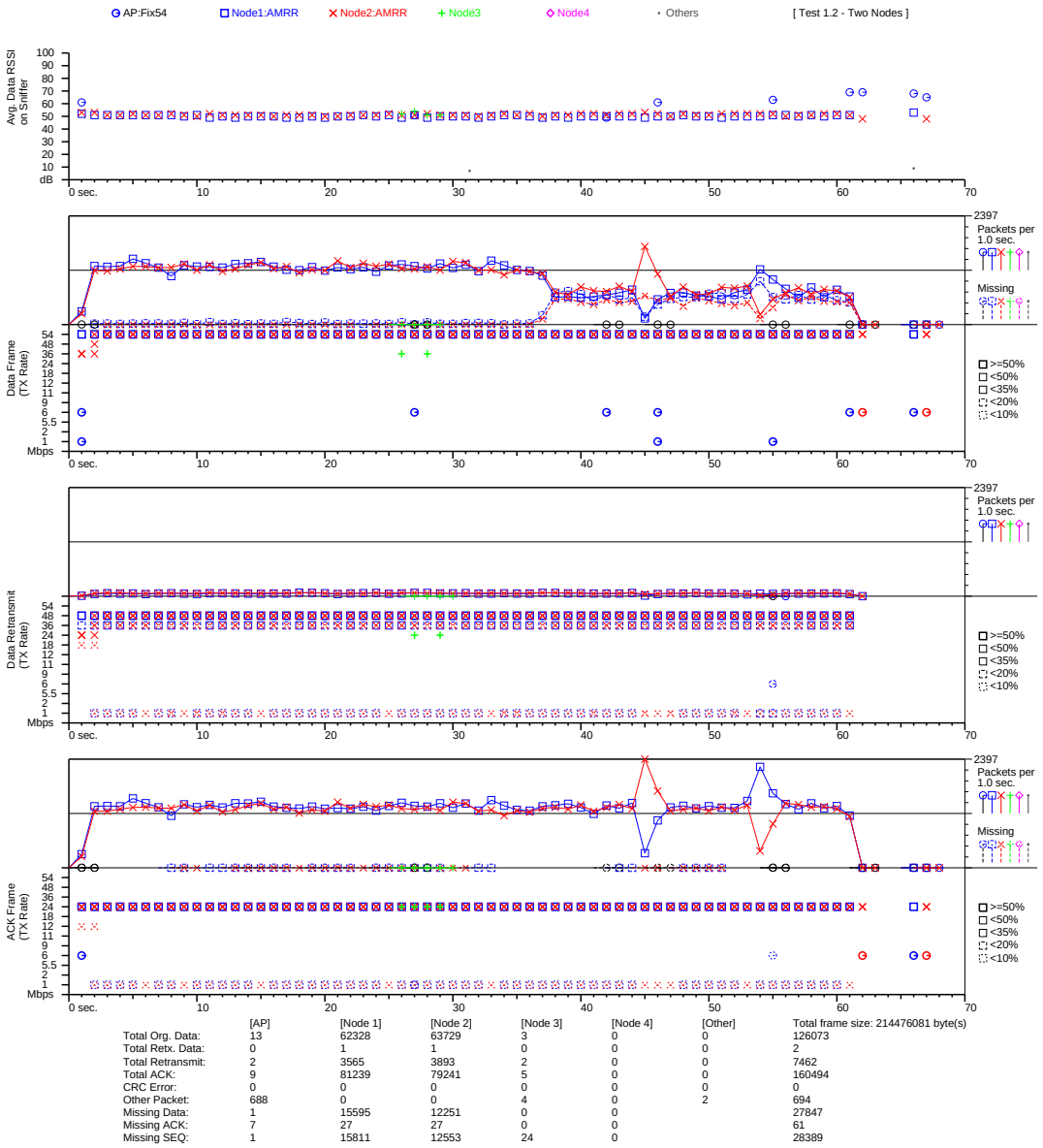
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 204792098 byte(s)
Total Org. Data:	5	0	6	125869	3	0	125883
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	2	0	0	2225	6	0	2233
Total ACK:	11	0	6	128037	4	0	128058
CRC Error:	0	0	0	0	0	0	0
Other Packet:	683	0	6	0	4	0	693
Missing Data:	5	0	0	346	0		351
Missing ACK:	1	0	0	65	0		66
Missing SEQ:	8	0	22	1734	24		1788

Figure B.10: Test 1.2 - Only Node 3 uses AMRR (Full Graph)



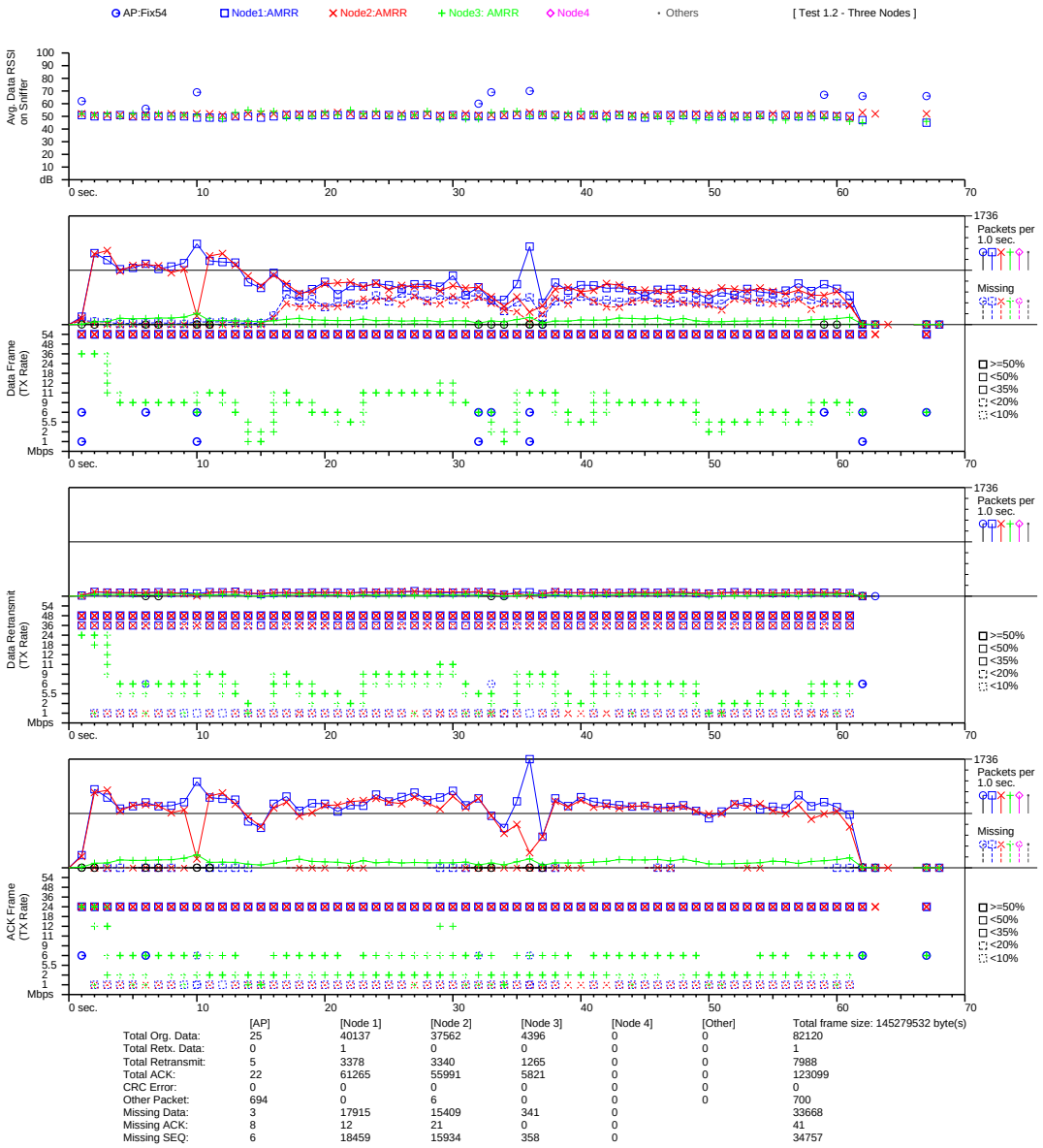
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 175960941 byte(s)
Total Org. Data:	6	4	0	0	97614	0	97624
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	3	1	0	0	12583	0	12587
Total ACK:	10	6	0	0	104187	0	104203
CRC Error:	0	0	0	0	0	0	0
Other Packet:	672	4	0	0	0	0	676
Missing Data:	3	1	0	0	4341		4345
Missing ACK:	0	0	0	0	432		432
Missing SEQ:	15	24	0	0	5736		5775

Figure B.11: Test 1.2 - Only Node 4 uses AMRR (Full Graph)



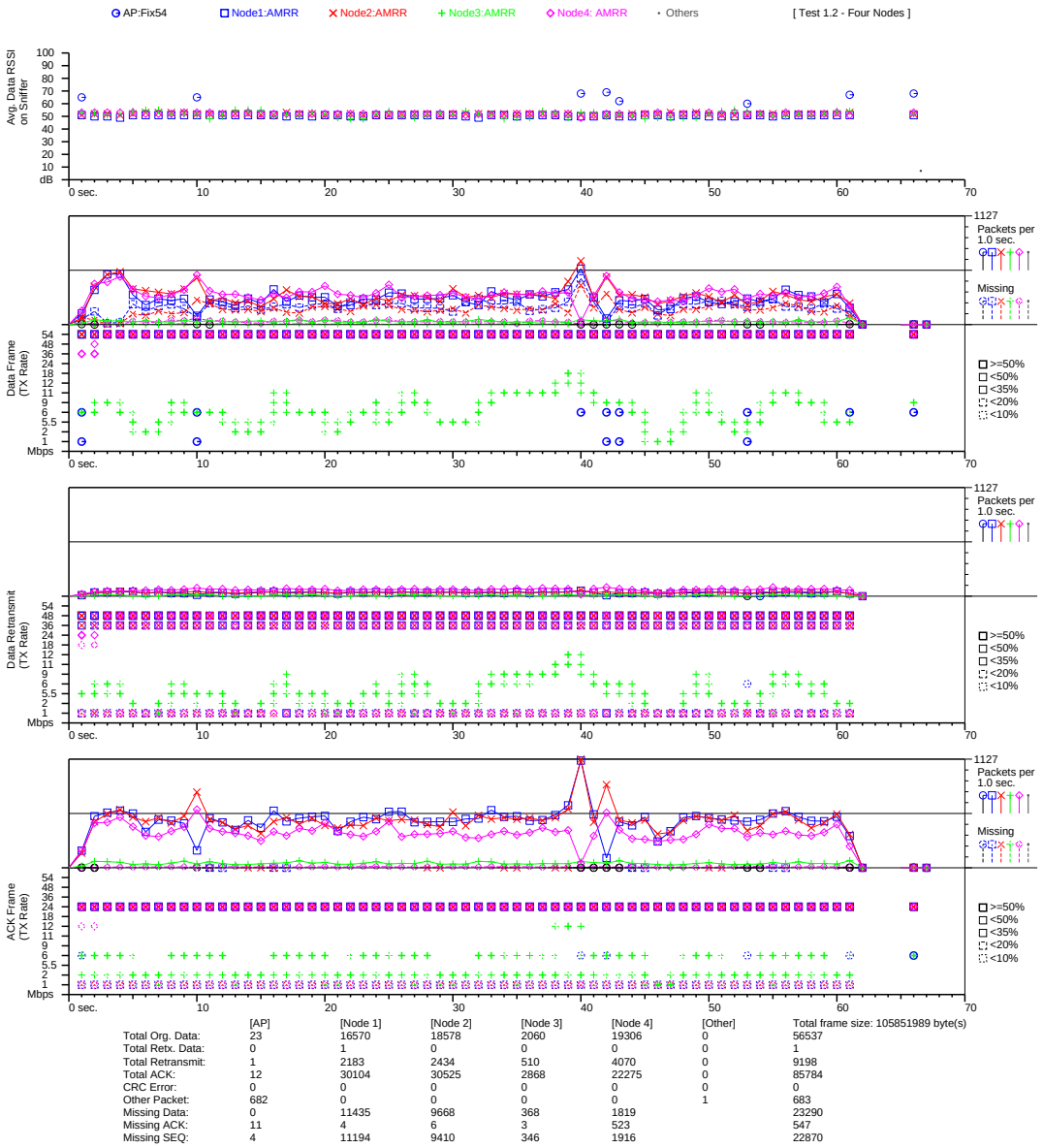
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 214476081 byte(s)
Total Org. Data:	13	62328	63729	3	0	0	126073
Total Retx. Data:	0	1	1	0	0	0	2
Total Retransmit:	2	3565	3893	2	0	0	7462
Total ACK:	9	81239	79241	5	0	0	160494
CRC Error:	0	0	0	0	0	0	0
Other Packet:	688	0	0	4	0	2	694
Missing Data:	1	15595	12251	0	0		27847
Missing ACK:	7	27	27	0	0		61
Missing SEQ:	1	15811	12553	24	0		28389

Figure B.12: Test 1.2 - Two active clients use AMRR (Full Graph)



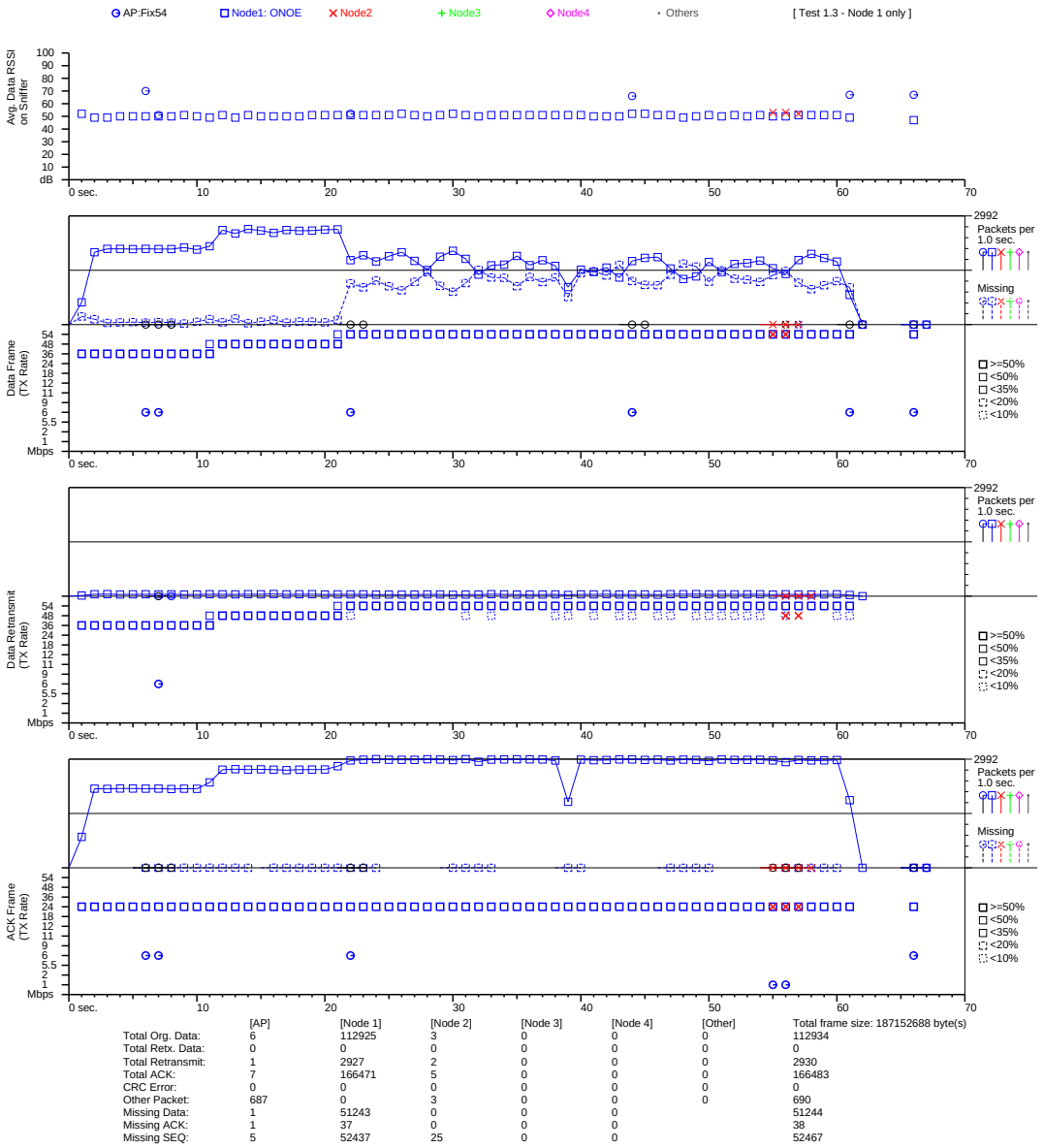
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 145279532 byte(s)
Total Org. Data:	25	40137	37562	4396	0	0	82120
Total Retx. Data:	0	1	0	0	0	0	1
Total Retransmit:	5	3378	3340	1265	0	0	7988
Total ACK:	22	61265	55991	5821	0	0	123099
CRC Error:	0	0	0	0	0	0	0
Other Packet:	694	0	6	0	0	0	700
Missing Data:	3	17915	15409	341	0		33668
Missing ACK:	8	12	21	0	0		41
Missing SEQ:	6	18459	15934	358	0		34757

Figure B.13: Test 1.2 - Three active clients use AMRR (Full Graph)



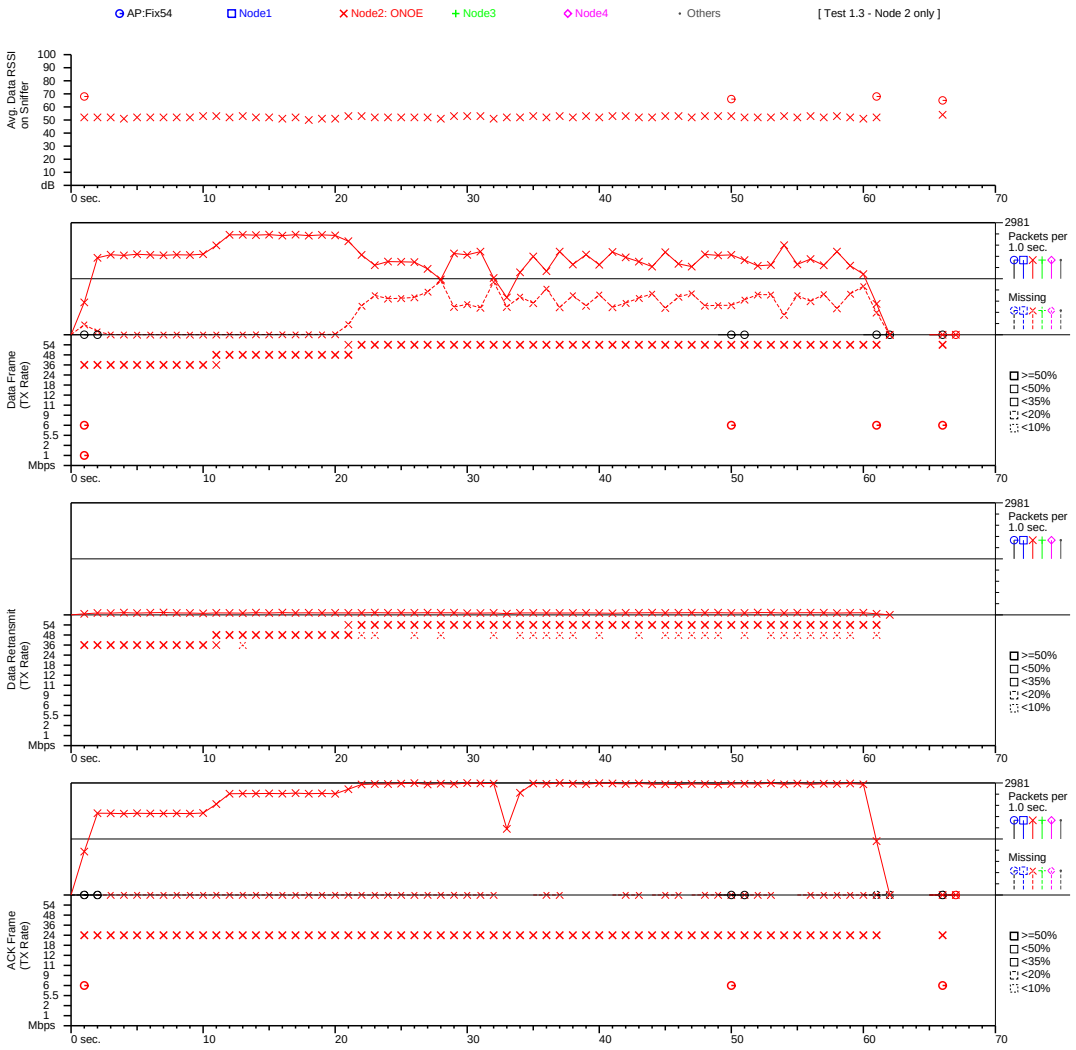
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 105851989 byte(s)
Total Org. Data:	23	16570	18578	2060	19306	0	56537
Total Retx. Data:	0	1	0	0	0	0	1
Total Retransmit:	1	2183	2434	510	4070	0	9198
Total ACK:	12	30104	30525	2868	22275	0	85784
CRC Error:	0	0	0	0	0	0	0
Other Packet:	682	0	0	0	0	1	683
Missing Data:	0	11435	9668	368	1819		23290
Missing ACK:	11	4	6	3	523		547
Missing SEQ:	4	11194	9410	346	1916		22870

Figure B.14: Test 1.2 - Four active clients use AMRR (Full Graph)



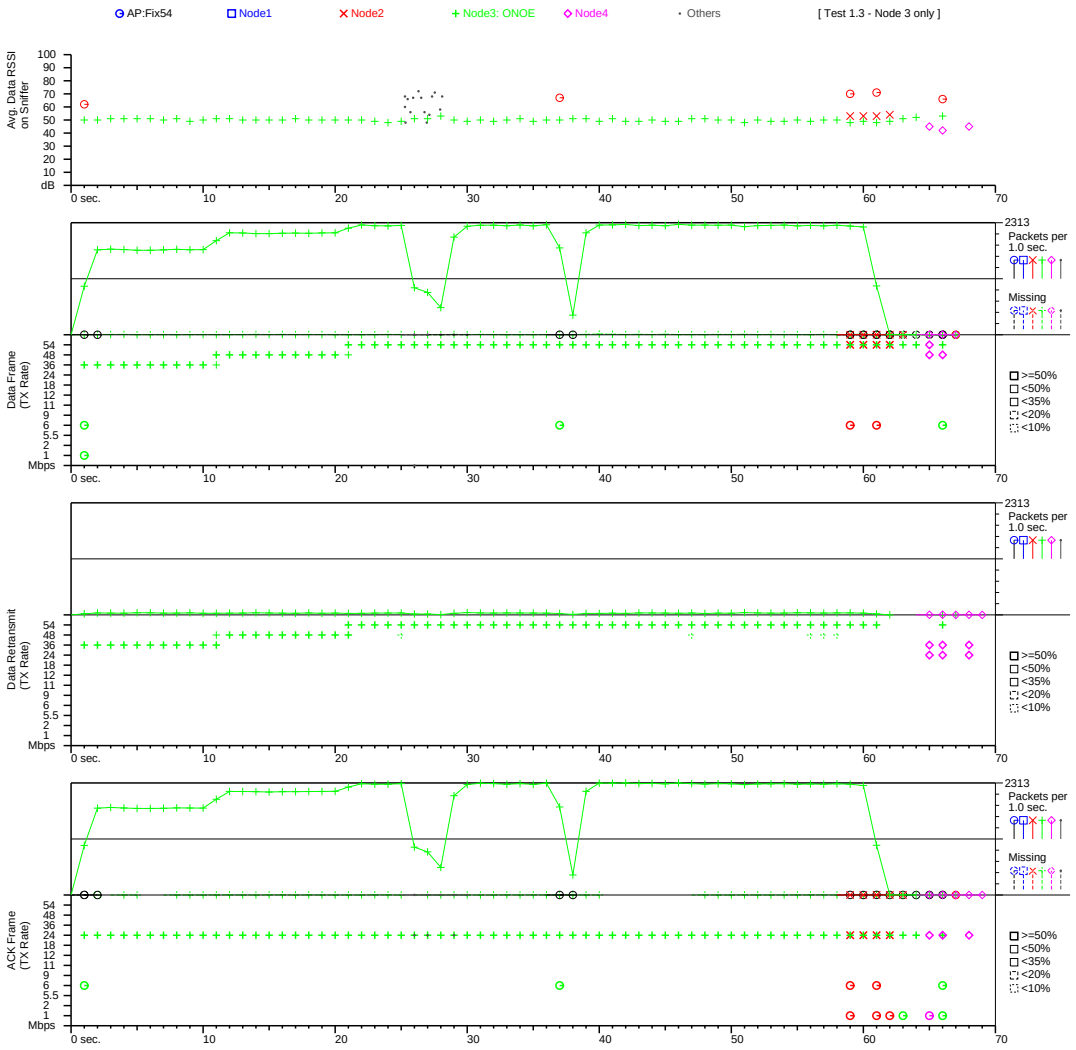
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	
Total Org. Data:	6	112925	3	0	0	0	112934
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	1	2927	2	0	0	0	2930
Total ACK:	7	166471	5	0	0	0	166483
CRC Error:	0	0	0	0	0	0	0
Other Packet:	687	0	3	0	0	0	690
Missing Data:	1	51243	0	0	0		51244
Missing ACK:	1	37	0	0	0		38
Missing SEQ:	5	52437	25	0	0		52467

Figure B.15: Test 1.3 - Only Node 1 uses ONOE (Full Graph)



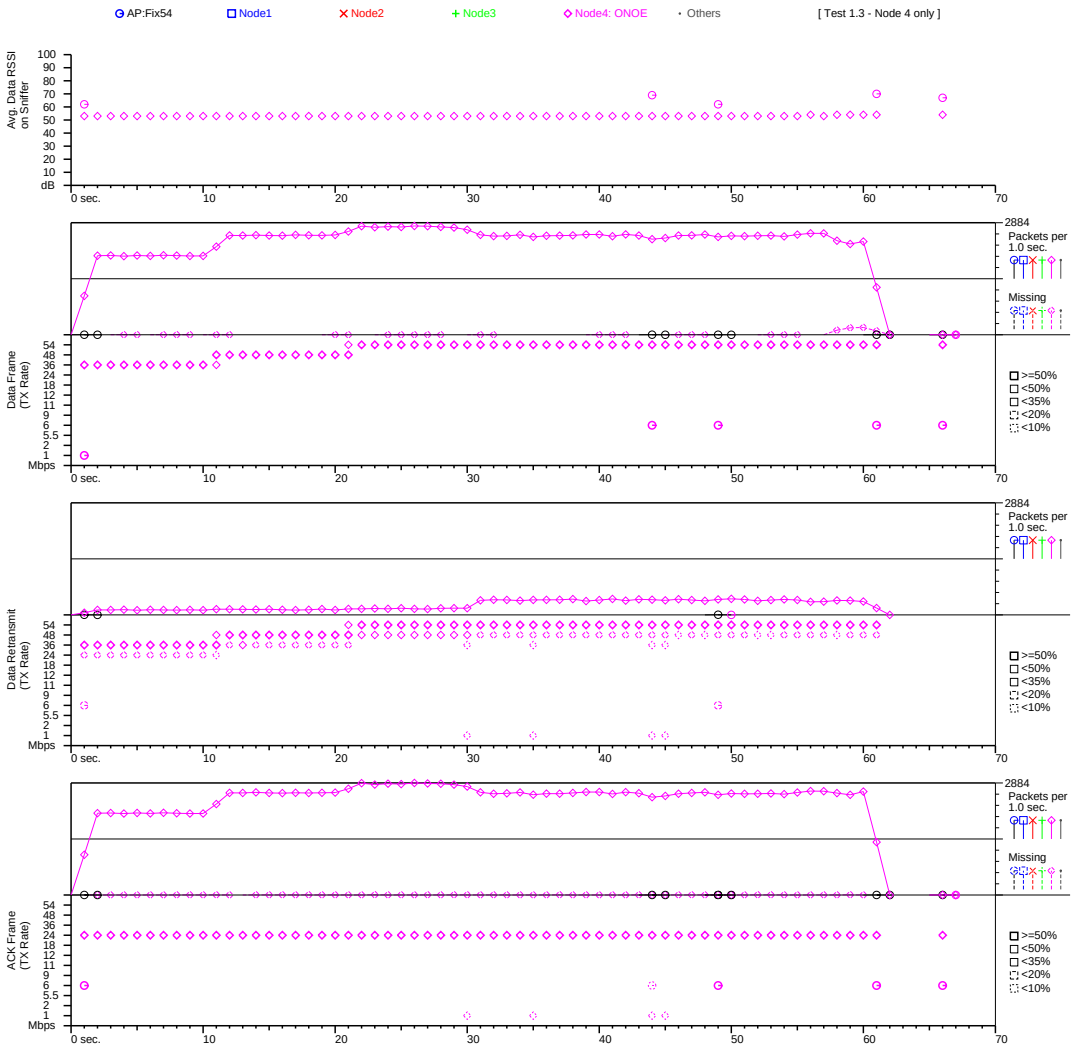
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 206005698 byte(s)
Total Org. Data:	6	0	124777	0	0	0	124783
Total Retx. Data:	0	0	1	0	0	0	1
Total Retransmit:	0	0	3184	0	0	0	3184
Total ACK:	3	0	165552	0	0	0	165555
CRC Error:	0	0	0	0	0	0	0
Other Packet:	678	0	0	0	0	0	678
Missing Data:	0	0	38221	0	0		38221
Missing ACK:	3	0	48	0	0		51
Missing SEQ:	6	0	40180	0	0		40186

Figure B.16: Test 1.3 - Only Node 2 uses ONOE (Full Graph)



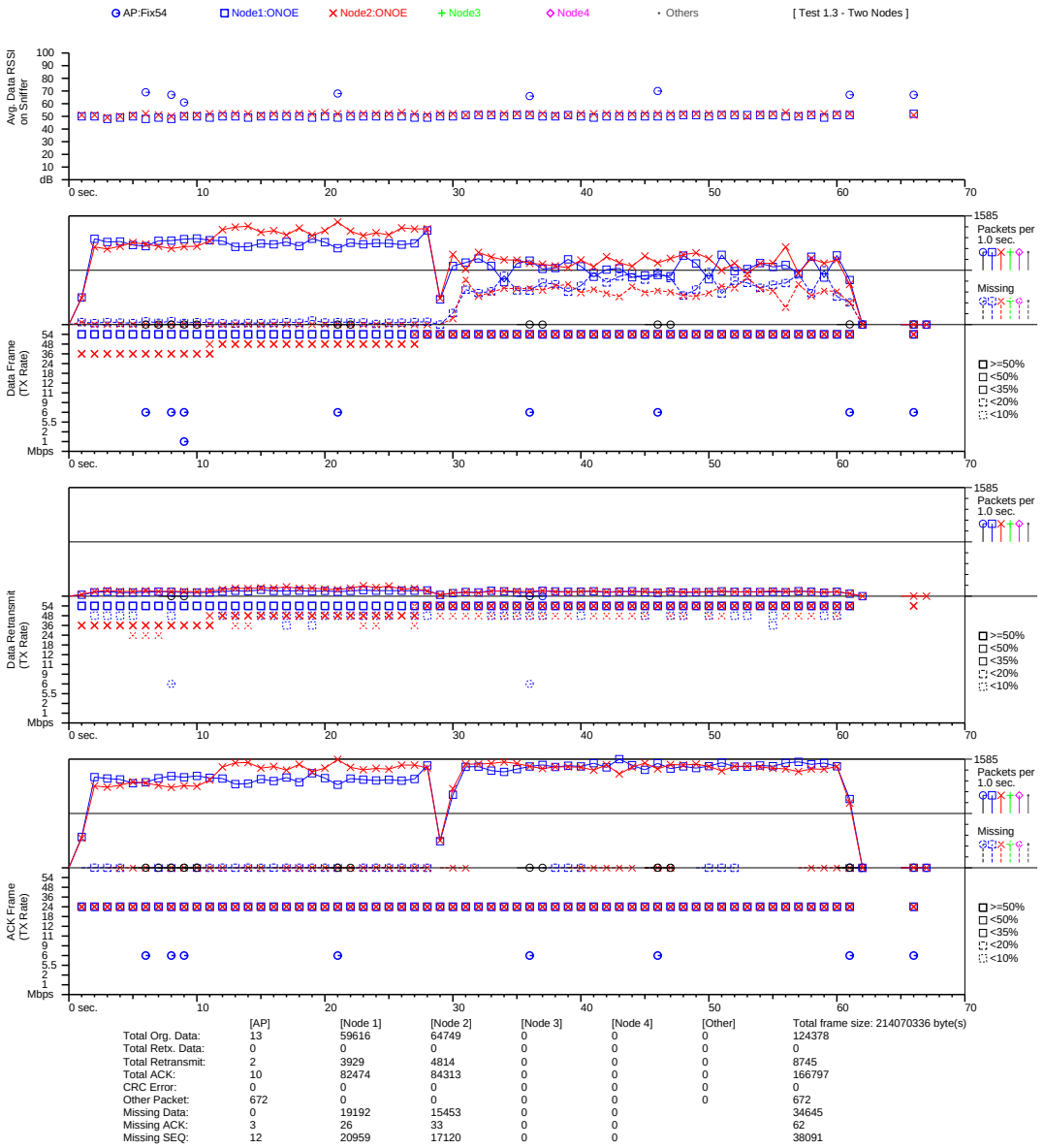
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 198089810 byte(s)
Total Org. Data:	6	0	5	121772	4	1	121788
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	0	0	0	2134	8	0	2142
Total ACK:	13	0	5	123910	5	4	123937
CRC Error:	0	0	0	0	0	0	0
Other Packet:	682	0	6	5	4	14	711
Missing Data:	8	0	0	290	0		298
Missing ACK:	1	0	0	64	0		65
Missing SEQ:	19	0	22	7603	24		7668

Figure B.17: Test 1.3 - Only Node 3 uses ONOE (Full Graph)



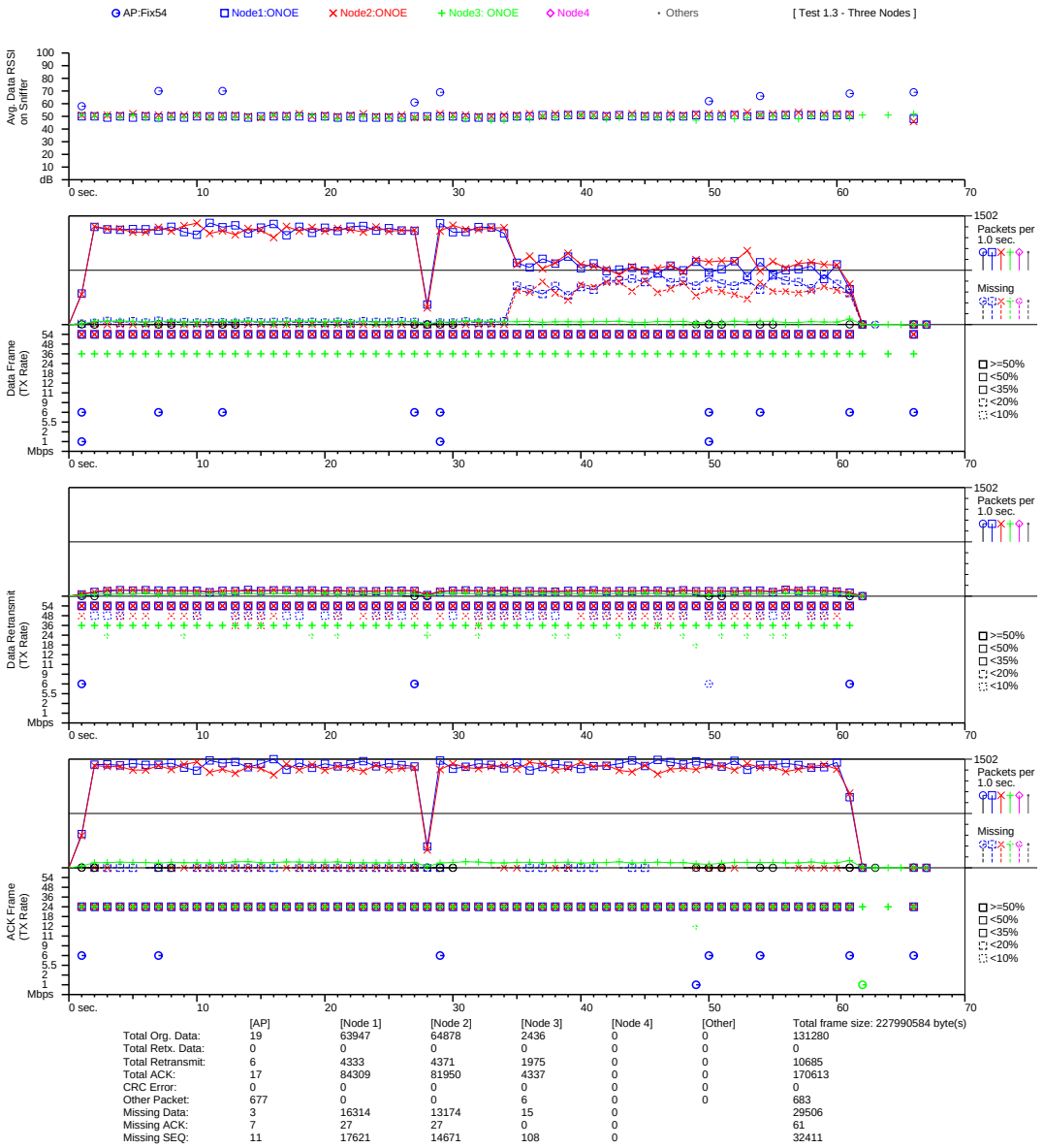
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 263823892 byte(s)
Total Org. Data:	6	0	0	0	149521	0	149527
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	6	0	0	0	15778	0	15784
Total ACK:	5	0	0	0	154280	0	154285
CRC Error:	0	0	0	0	0	0	0
Other Packet:	679	0	0	0	0	0	679
Missing Data:	0	0	0	0	601		601
Missing ACK:	1	0	0	0	95		96
Missing SEQ:	6	0	0	0	1668		1674

Figure B.18: Test 1.3 - Only Node 4 uses ONOE (Full Graph)



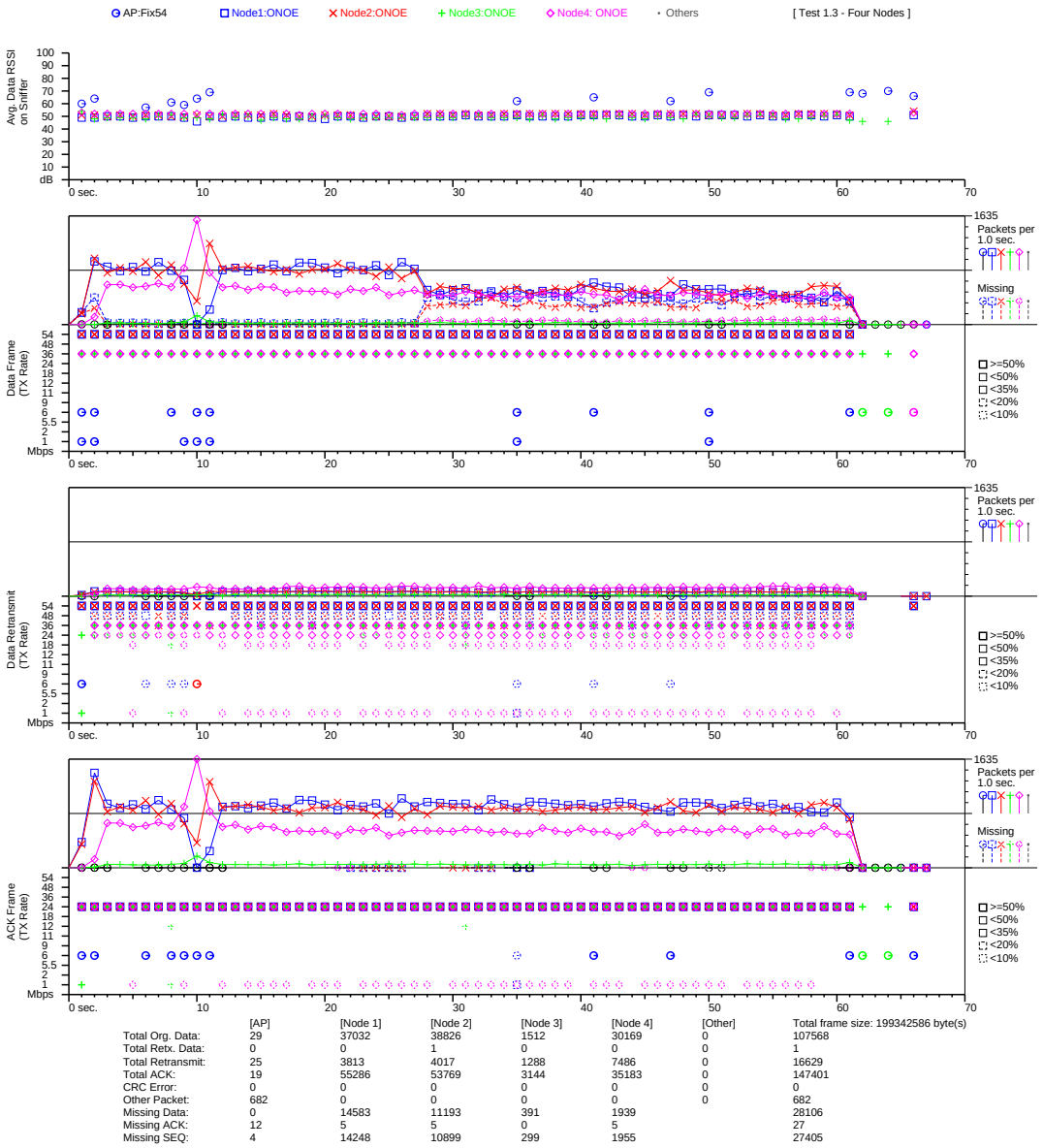
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 214070336 byte(s)
Total Org. Data:	13	59616	64749	0	0	0	124378
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	2	3929	4814	0	0	0	8745
Total ACK:	10	82474	84313	0	0	0	166797
CRC Error:	0	0	0	0	0	0	0
Other Packet:	672	0	0	0	0	0	672
Missing Data:	0	19192	15453	0	0		34645
Missing ACK:	3	26	33	0	0		62
Missing SEQ:	12	20959	17120	0	0		38091

Figure B.19: Test 1.3 - Two active clients use ONOE (Full Graph)



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 227990584 byte(s)
Total Org. Data:	19	63947	64878	2436	0	0	131280
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	6	4333	4371	1975	0	0	10685
Total ACK:	17	84309	81950	4337	0	0	170613
CRC Error:	0	0	0	0	0	0	0
Other Packet:	677	0	0	6	0	0	683
Missing Data:	3	16314	13174	15	0		29506
Missing ACK:	7	27	27	0	0		61
Missing SEQ:	11	17621	14671	108	0		32411

Figure B.20: Test 1.3 - Three active clients use ONOE (Full Graph)



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 199342586 byte(s)
Total Org. Data:	29	37032	38826	1512	30169	0	107568
Total Retx. Data:	0	0	1	0	0	0	1
Total Retransmit:	25	3813	4017	1288	7486	0	16629
Total ACK:	19	55286	53769	3144	35183	0	147401
CRC Error:	0	0	0	0	0	0	0
Other Packet:	682	0	0	0	0	0	682
Missing Data:	0	14583	11193	391	1939		28106
Missing ACK:	12	5	5	0	5		27
Missing SEQ:	4	14248	10899	299	1955		27405

Figure B.21: Test 1.3 - Four active clients use ONOE (Full Graph)

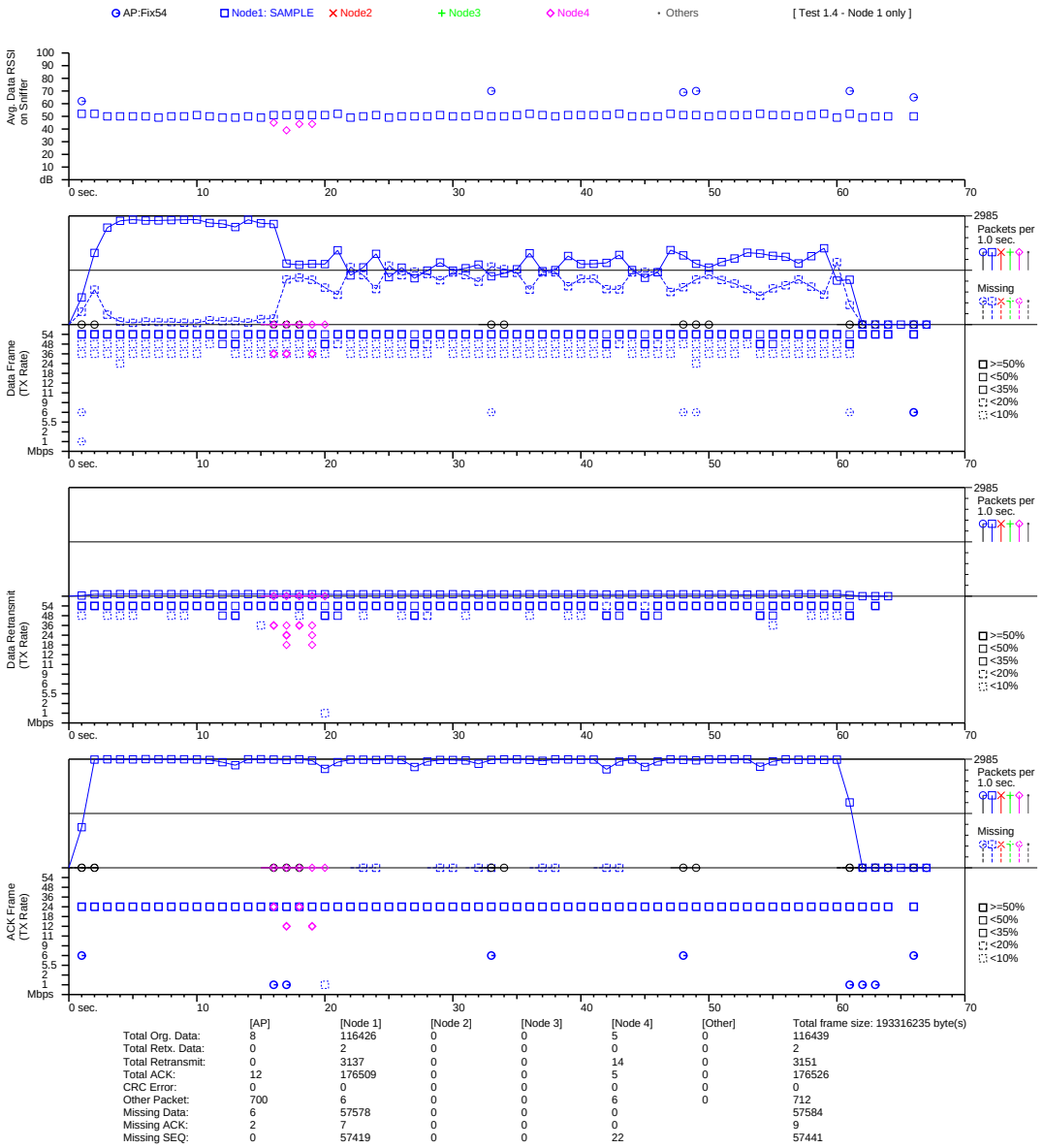
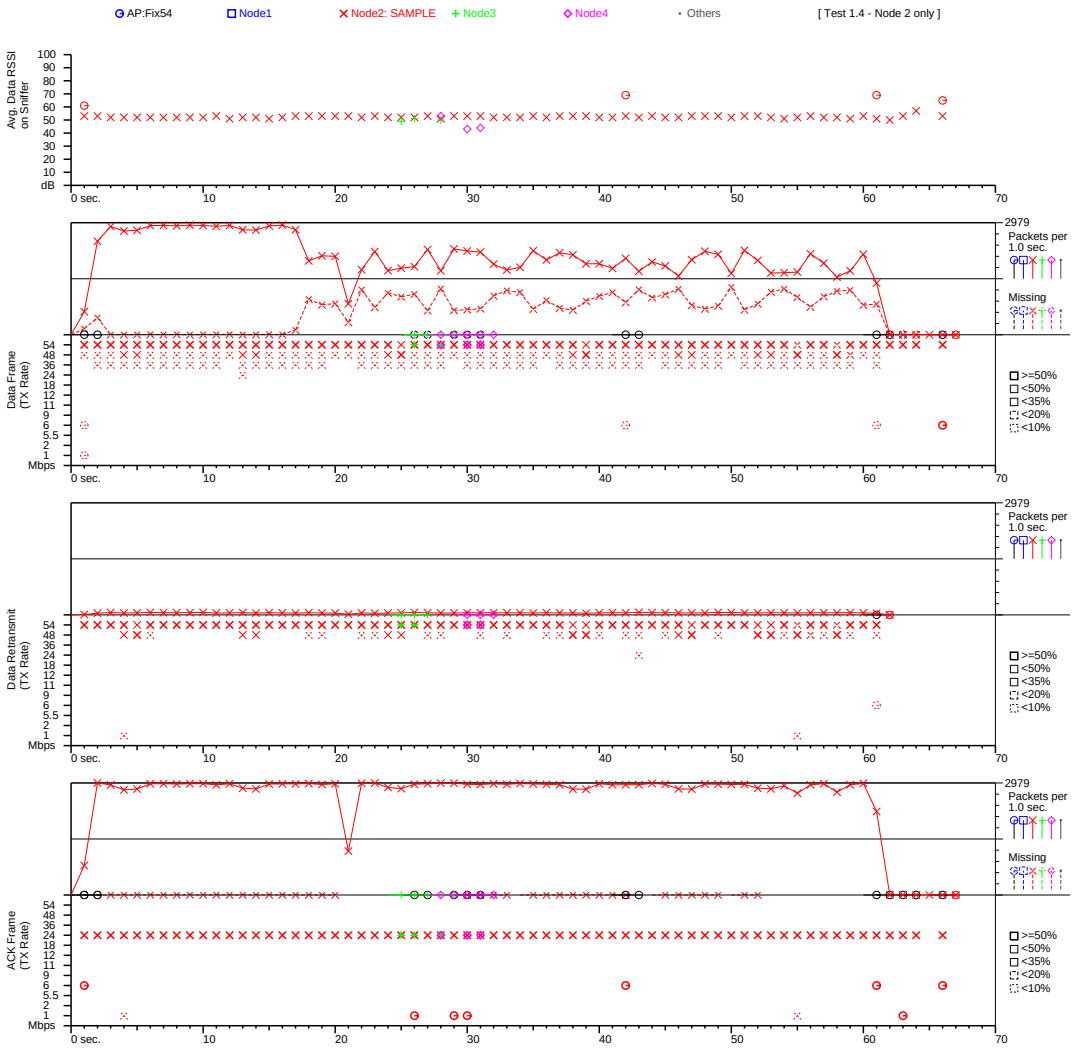
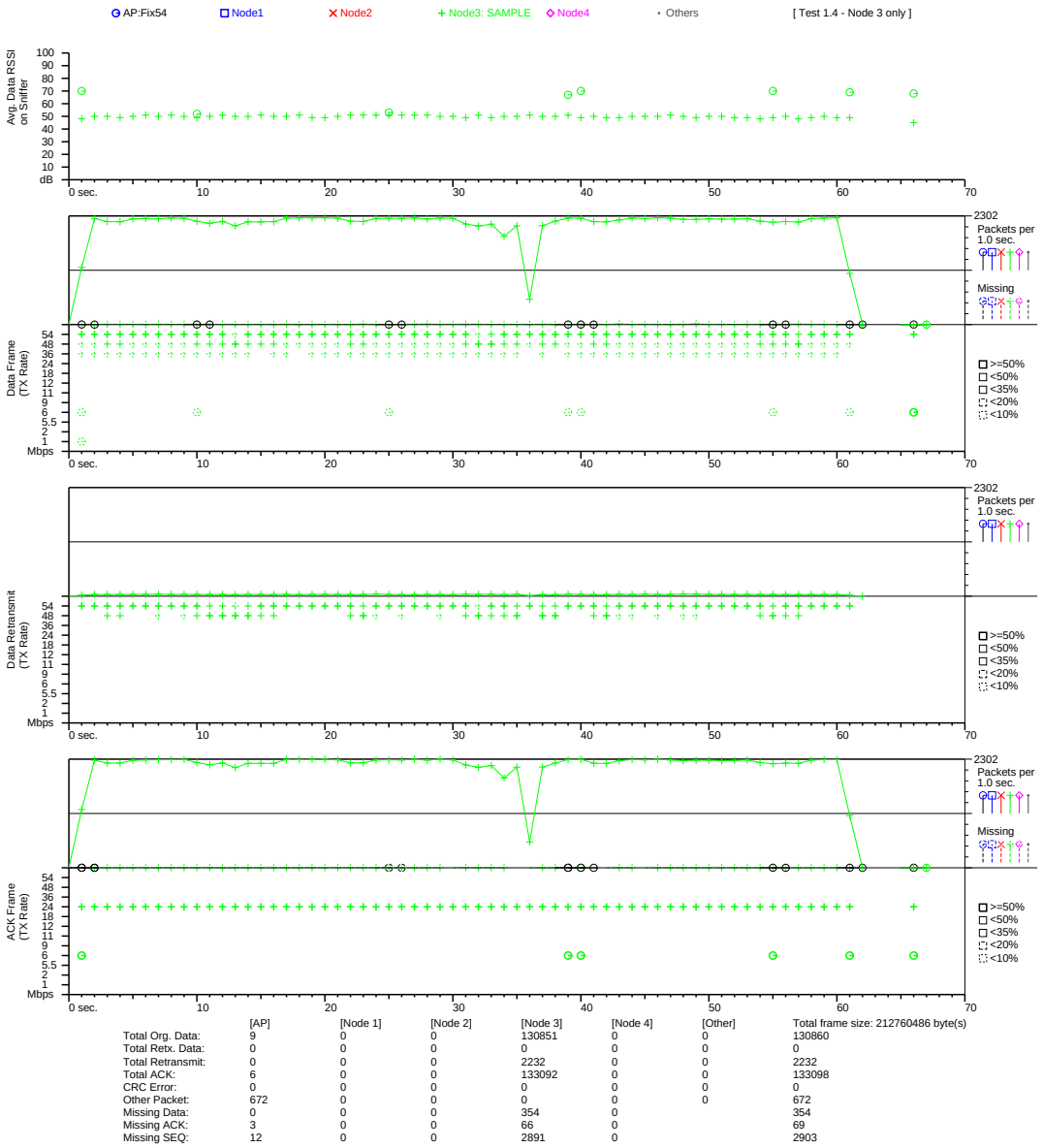


Figure B.22: Test 1.4 - Only Node 1 uses SAMPLE (Full Graph)



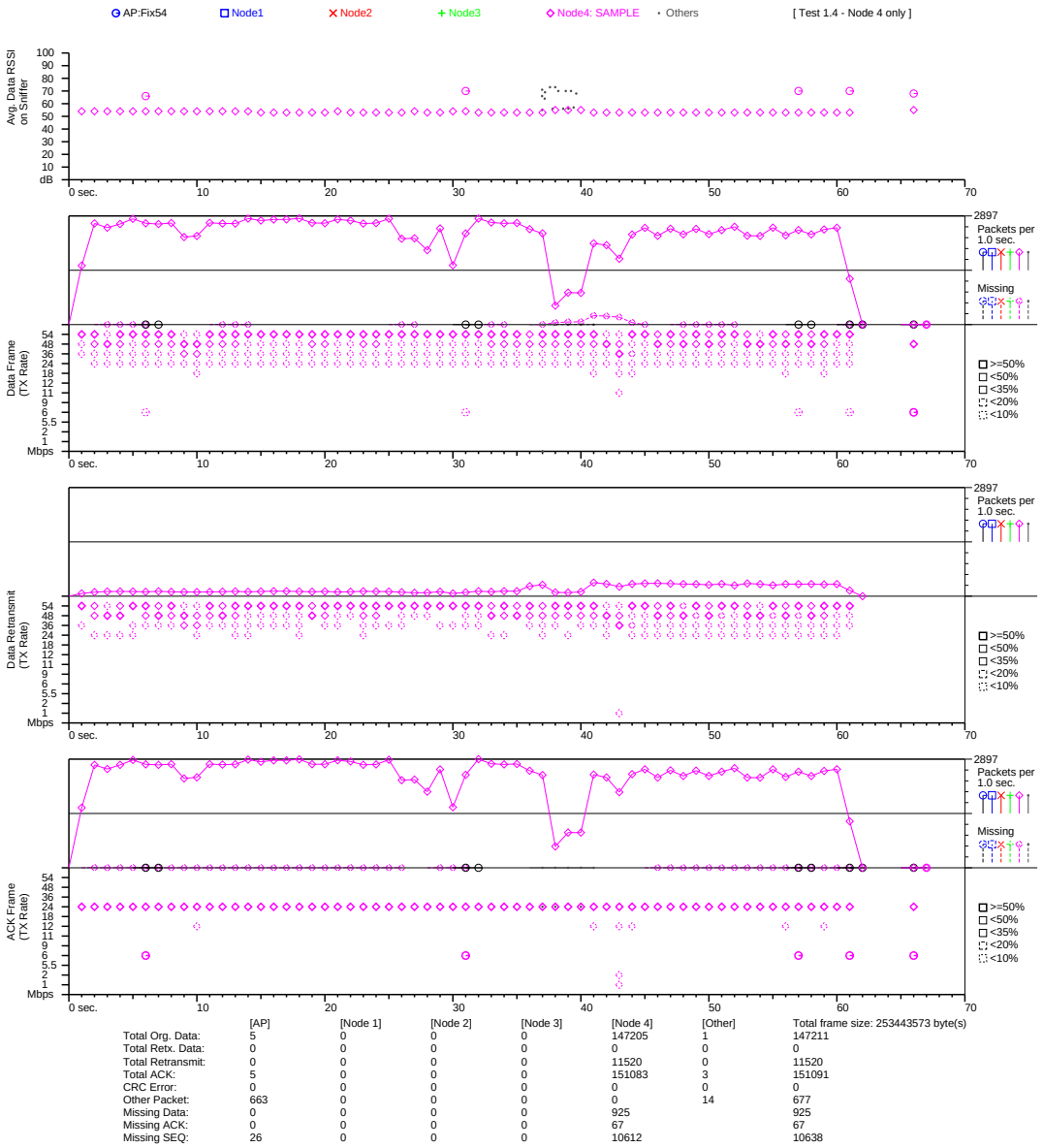
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 213549668 byte(s)
Total Org. Data:	5	0	129361	2	4	0	129372
Total Retx. Data:	0	0	1	0	0	0	1
Total Retransmit:	1	0	3249	3	4	0	3257
Total ACK:	10	0	173294	5	4	0	173313
CRC Error:	0	0	0	0	0	0	0
Other Packet:	684	0	6	6	5	0	701
Missing Data:	6	0	41433	0	0		41439
Missing ACK:	1	0	61	0	0		62
Missing SEQ:	10	0	43804	22	23		43859

Figure B.23: Test 1.4 - Only Node 2 uses SAMPLE (Full Graph)



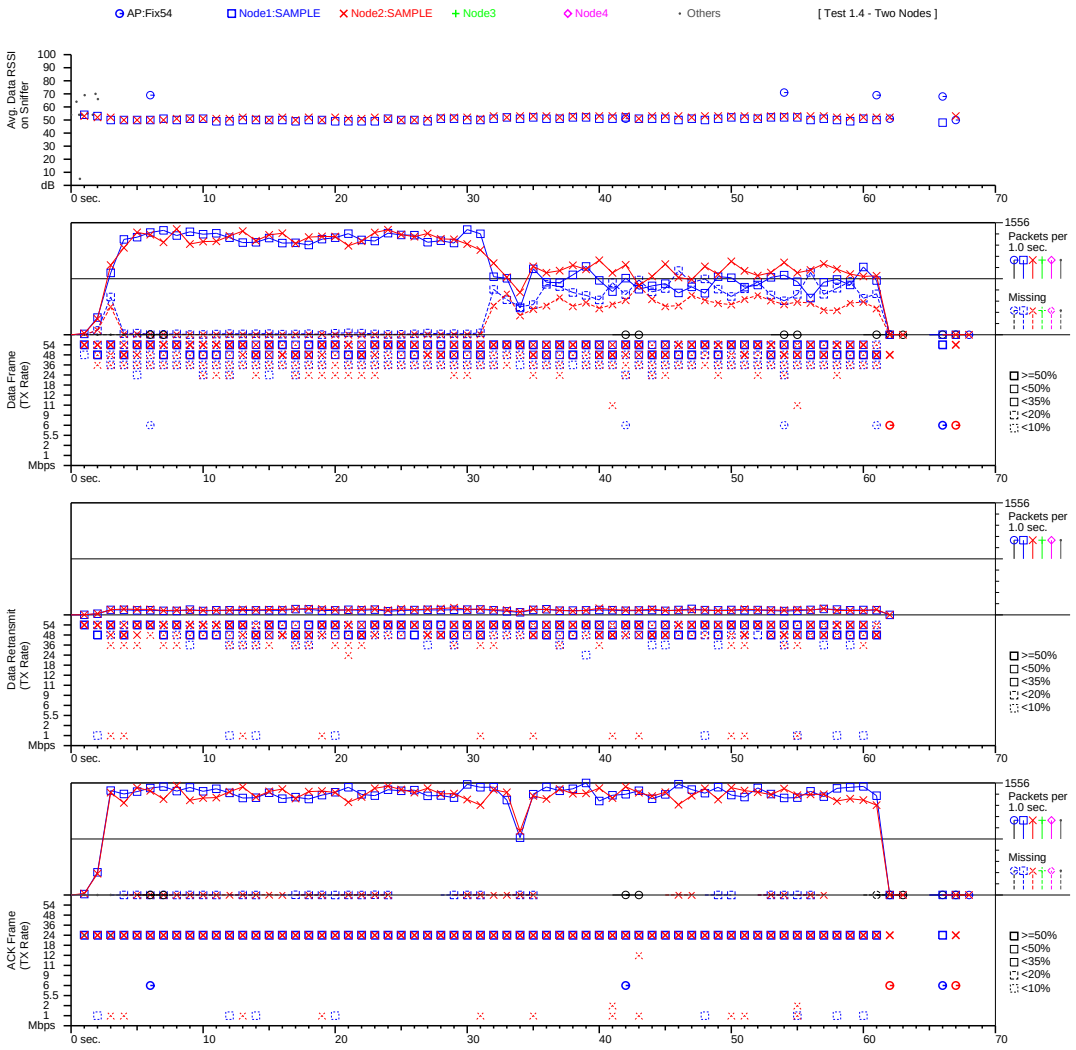
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 212760486 byte(s)
Total Org. Data:	9	0	0	130851	0	0	130860
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	0	0	0	2232	0	0	2232
Total ACK:	6	0	0	133092	0	0	133098
CRC Error:	0	0	0	0	0	0	0
Other Packet:	672	0	0	0	0	0	672
Missing Data:	0	0	0	354	0		354
Missing ACK:	3	0	0	66	0		69
Missing SEQ:	12	0	0	2891	0		2903

Figure B.24: Test 1.4 - Only Node 3 uses SAMPLE (Full Graph)



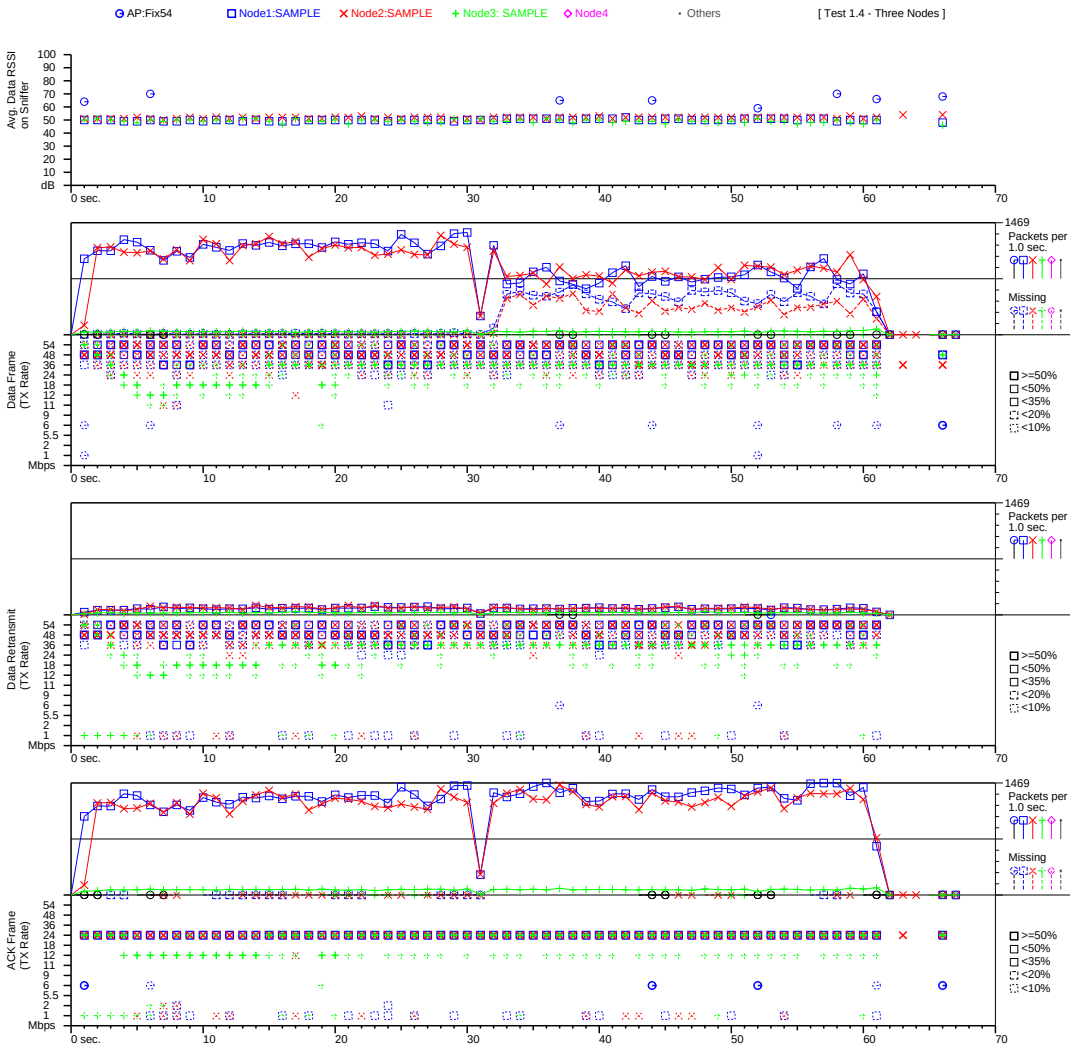
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 253443573 byte(s)
Total Org. Data:	5	0	0	0	147205	1	147211
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	0	0	0	0	11520	0	11520
Total ACK:	5	0	0	0	151083	3	151091
CRC Error:	0	0	0	0	0	0	0
Other Packet:	663	0	0	0	0	14	677
Missing Data:	0	0	0	0	925		925
Missing ACK:	0	0	0	0	67		67
Missing SEQ:	26	0	0	0	10612		10638

Figure B.25: Test 1.4 - Only Node 4 uses SAMPLE (Full Graph)



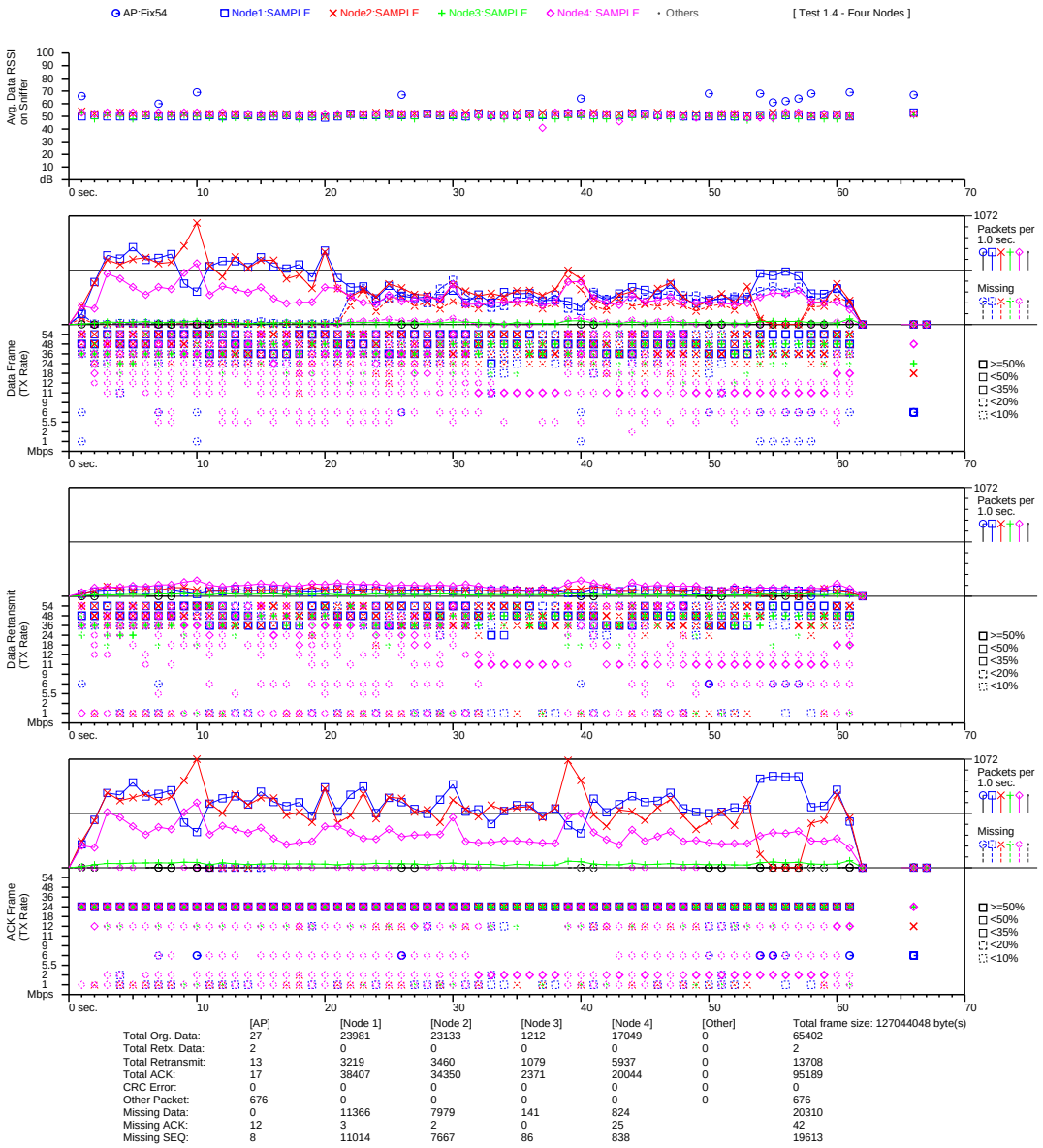
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 215357492 byte(s)
Total Org. Data:	7	60733	65128	0	0	0	125868
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	0	3846	4233	0	0	0	8079
Total ACK:	5	83802	82474	0	0	3	166284
CRC Error:	0	0	0	0	0	0	0
Other Packet:	671	0	0	0	0	11	682
Missing Data:	0	19559	13783	0	0		33342
Missing ACK:	2	21	34	0	0		57
Missing SEQ:	17	21530	15843	0	0		37390

Figure B.26: Test 1.4 - Two active clients use SAMPLE (Full Graph)



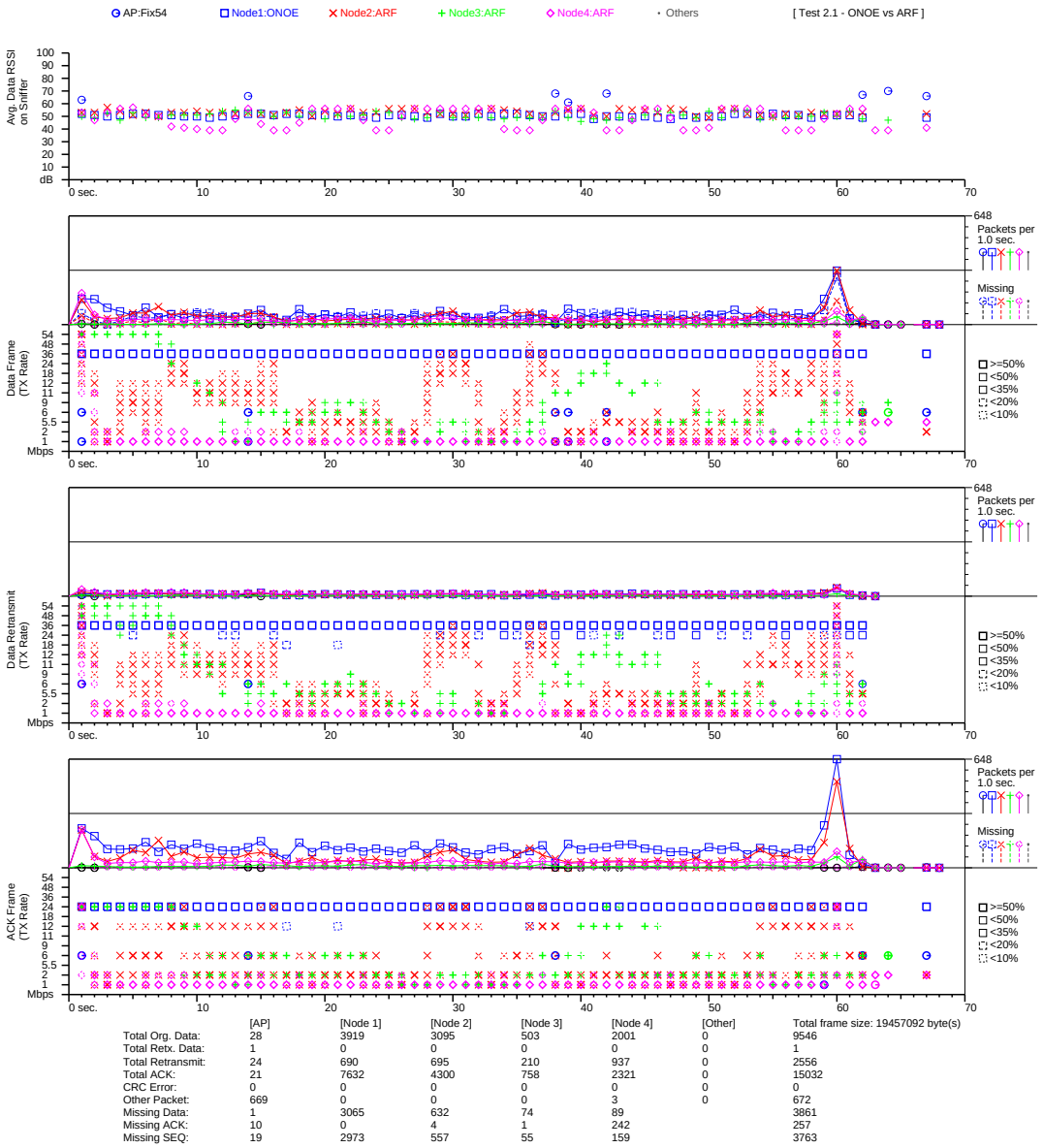
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 208072702 byte(s)
Total Org. Data:	15	57755	57322	2546	0	0	117638
Total Retx. Data:	0	2	0	0	0	0	2
Total Retransmit:	7	4980	5212	1732	0	0	11931
Total ACK:	13	77657	73254	4237	0	0	155161
CRC Error:	0	0	0	0	0	0	0
Other Packet:	686	0	2	0	0	0	688
Missing Data:	0	15216	11224	58	0		26498
Missing ACK:	4	13	21	4	0		42
Missing SEQ:	10	16308	12396	107	0		28821

Figure B.27: Test 1.4 - Three active clients use SAMPLE (Full Graph)



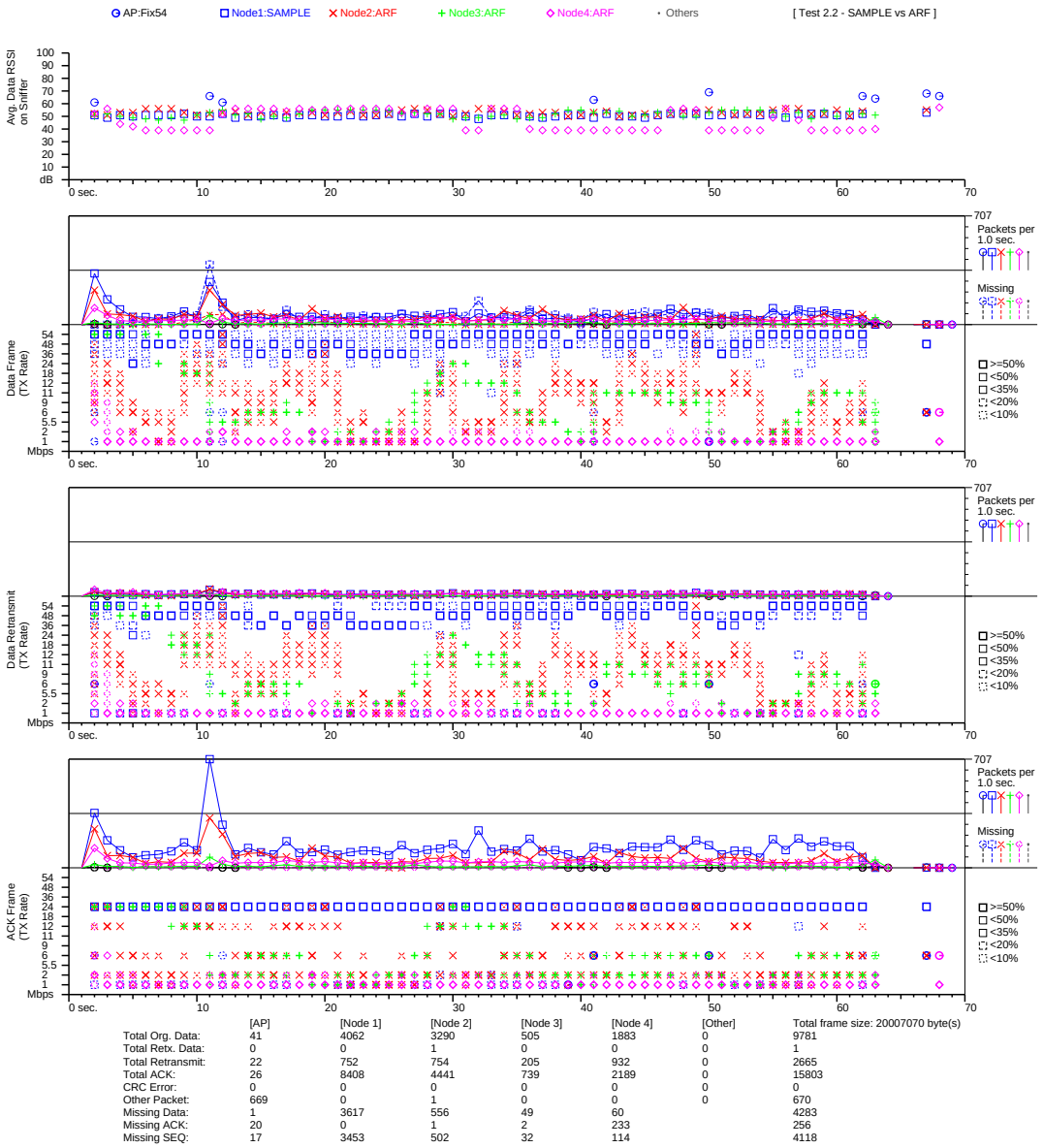
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 127044048 byte(s)
Total Org. Data:	27	23981	23133	1212	17049	0	65402
Total Retx. Data:	2	0	0	0	0	0	2
Total Retransmit:	13	3219	3460	1079	5937	0	13708
Total ACK:	17	38407	34350	2371	20044	0	95189
CRC Error:	0	0	0	0	0	0	0
Other Packet:	676	0	0	0	0	0	676
Missing Data:	0	11366	7979	141	824		20310
Missing ACK:	12	3	2	0	25		42
Missing SEQ:	8	11014	7667	86	838		19613

Figure B.28: Test 1.4 - Four active clients use SAMPLE (Full Graph)



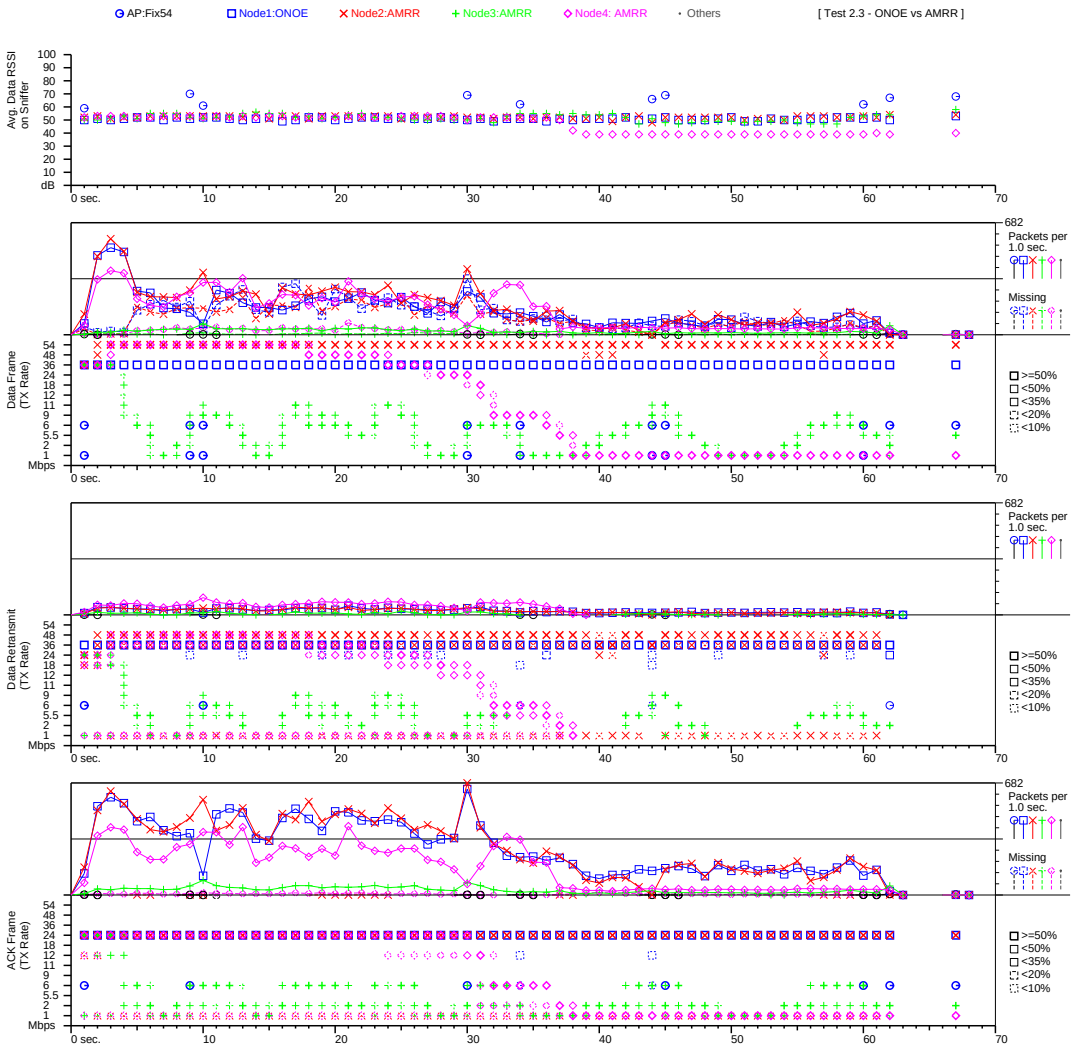
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 19457092 byte(s)
Total Org. Data:	28	3919	3095	503	2001	0	9546
Total Retx. Data:	1	0	0	0	0	0	1
Total Retransmit:	24	690	695	210	937	0	2556
Total ACK:	21	7632	4300	758	2321	0	15032
CRC Error:	0	0	0	0	0	0	0
Other Packet:	669	0	0	0	3	0	672
Missing Data:	1	3065	632	74	89		3861
Missing ACK:	10	0	4	1	242		257
Missing SEQ:	19	2973	557	55	159		3763

Figure B.29: Test 2.1 - ONOE against ARF (Full Graph)



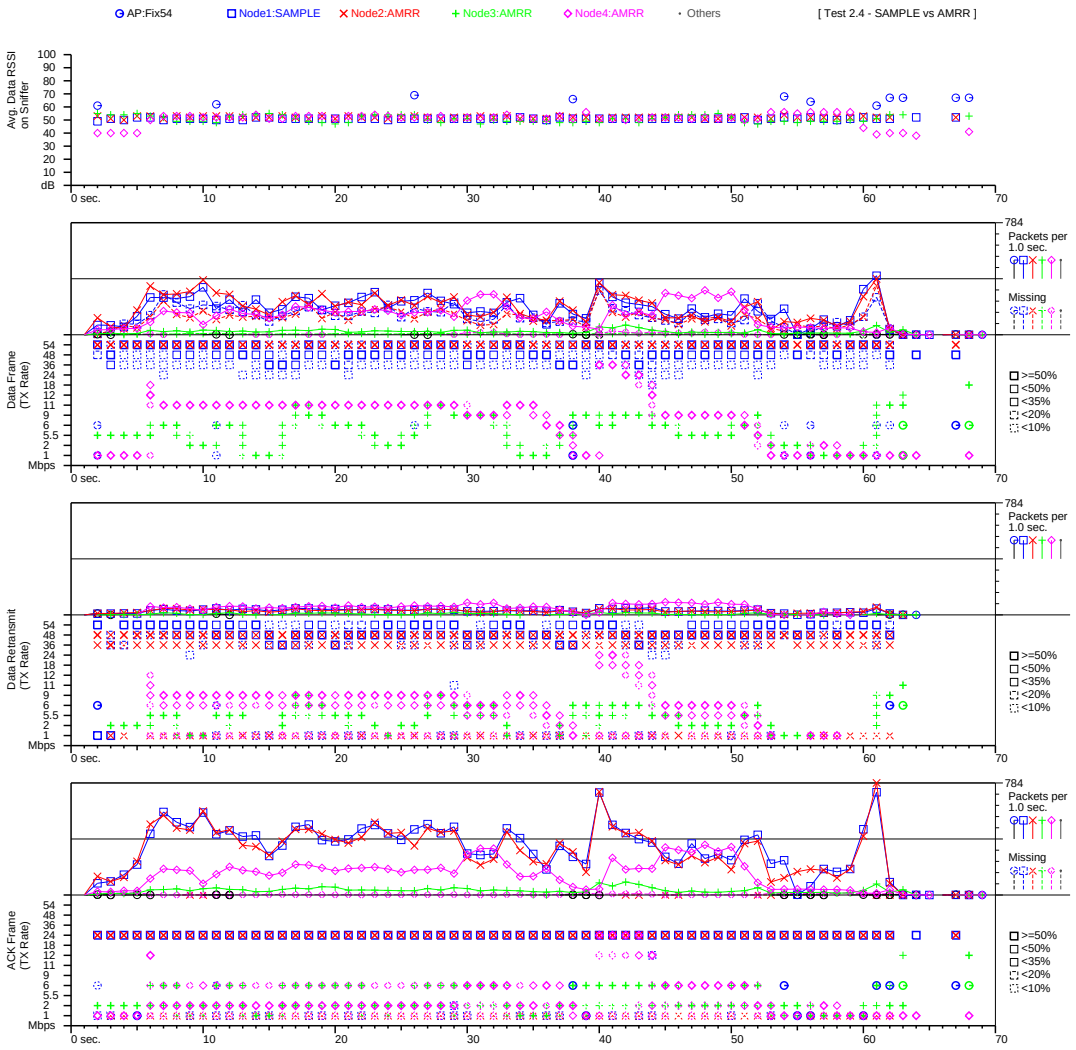
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 20007070 byte(s)
Total Org. Data:	41	4062	3290	505	1883	0	9781
Total Retx. Data:	0	0	1	0	0	0	1
Total Retransmit:	22	752	754	205	932	0	2665
Total ACK:	26	8408	4441	739	2189	0	15803
CRC Error:	0	0	0	0	0	0	0
Other Packet:	669	0	1	0	0	0	670
Missing Data:	1	3617	556	49	60		4283
Missing ACK:	20	0	1	2	233		256
Missing SEQ:	17	3453	502	32	114		4118

Figure B.30: Test 2.2 - SAMPLE against ARF (Full Graph)



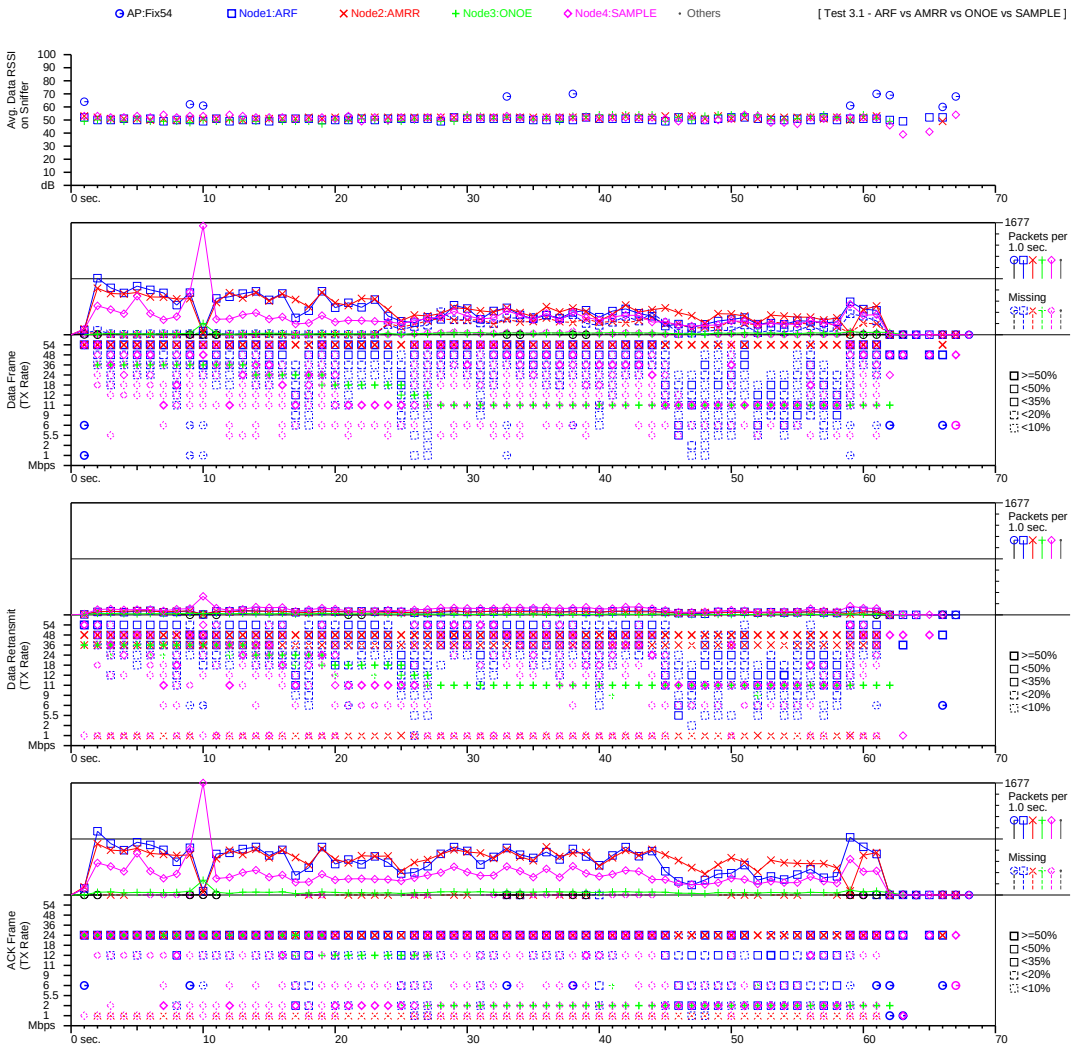
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 59661844 byte(s)
Total Org. Data:	34	9441	11107	1444	9209	0	31235
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	12	1564	1515	354	2333	0	5778
Total ACK:	19	18370	18974	1954	10674	0	49991
CRC Error:	0	0	0	0	0	0	0
Other Packet:	684	0	0	0	0	0	684
Missing Data:	0	7463	6441	221	963		15088
Missing ACK:	16	0	9	1	411		437
Missing SEQ:	1	7236	6179	203	1292		14911

Figure B.31: Test 2.3 - ONOE against AMRR (Full Graph)



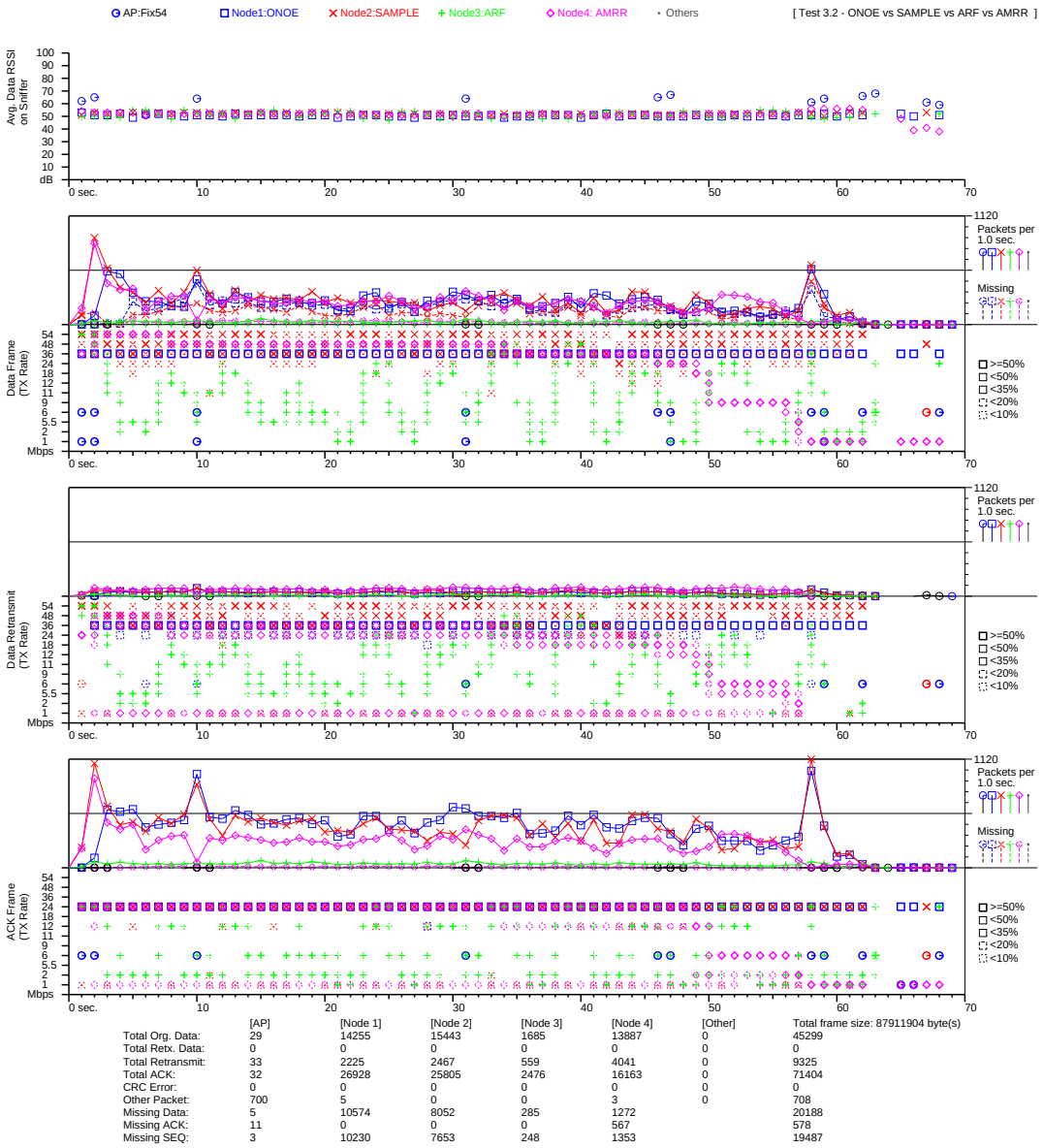
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 64057470 byte(s)
Total Org. Data:	32	11486	11731	1471	8439	0	33159
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	21	1778	1665	355	2790	0	6609
Total ACK:	23	21352	20596	2121	9388	0	53480
CRC Error:	0	0	0	0	0	0	0
Other Packet:	686	3	0	0	8	0	697
Missing Data:	4	8149	7284	343	234		16014
Missing ACK:	15	0	5	1	286		307
Missing SEQ:	9	7928	7092	303	329		15661

Figure B.32: Test 2.4 - SAMPLE against AMRR (Full Graph)



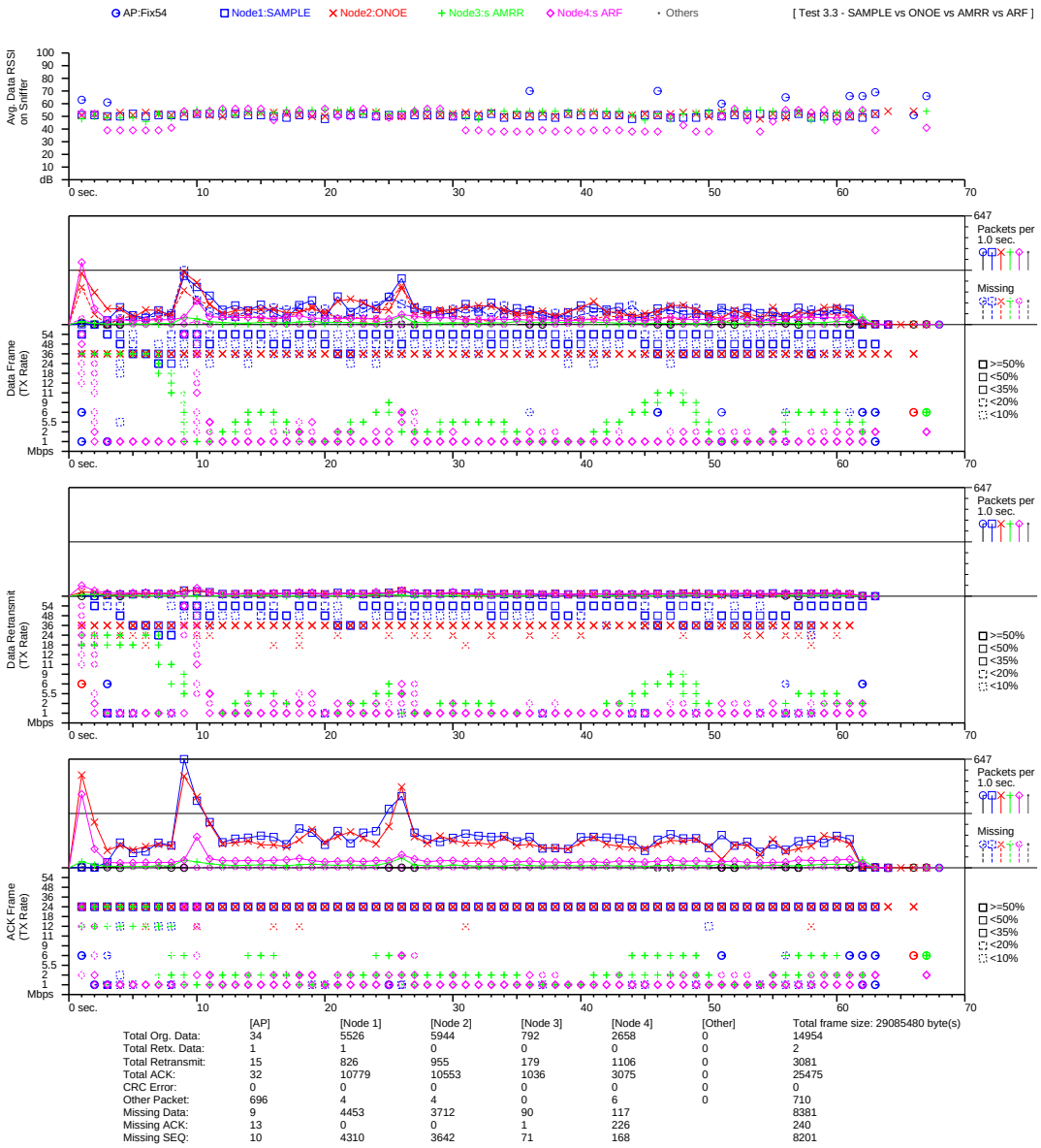
	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 121015740 byte(s)
Total Org. Data:	23	22134	23687	1468	16774	0	64086
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	10	2791	2672	758	5145	0	11376
Total ACK:	23	31409	33114	2561	19415	0	86522
CRC Error:	0	0	0	0	0	0	0
Other Packet:	682	4	0	0	5	0	691
Missing Data:	5	6652	6961	392	700		14710
Missing ACK:	10	4	16	0	13		43
Missing SEQ:	11	6396	6761	333	717		14218

Figure B.33: Test 3.1 - ARF, AMRR, ONOE, SAMPLE (Full Graph)



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 87911904 byte(s)
Total Org. Data:	29	14255	15443	1685	13887	0	45299
Total Retx. Data:	0	0	0	0	0	0	0
Total Retransmit:	33	2225	2467	559	4041	0	9325
Total ACK:	32	26928	25805	2476	16163	0	71404
CRC Error:	0	0	0	0	0	0	0
Other Packet:	700	5	0	0	3	0	708
Missing Data:	5	10574	8052	285	1272		20188
Missing ACK:	11	0	0	0	567		578
Missing SEQ:	3	10230	7653	248	1353		19487

Figure B.34: Test 3.2 - ONOE, SAMPLE, ARF, AMRR (Full Graph)



	[AP]	[Node 1]	[Node 2]	[Node 3]	[Node 4]	[Other]	Total frame size: 29085480 byte(s)
Total Org. Data:	34	5526	5944	792	2658	0	14954
Total Retx. Data:	1	1	0	0	0	0	2
Total Retransmit:	15	826	955	179	1106	0	3081
Total ACK:	32	10779	10553	1036	3075	0	25475
CRC Error:	0	0	0	0	0	0	0
Other Packet:	696	4	4	0	6	0	710
Missing Data:	9	4453	3712	90	117		8381
Missing ACK:	13	0	0	1	226		240
Missing SEQ:	10	4310	3642	71	168		8201

Figure B.35: Test 3.3 - SAMPLE, ONOE, AMRR, ARF (Full Graph)

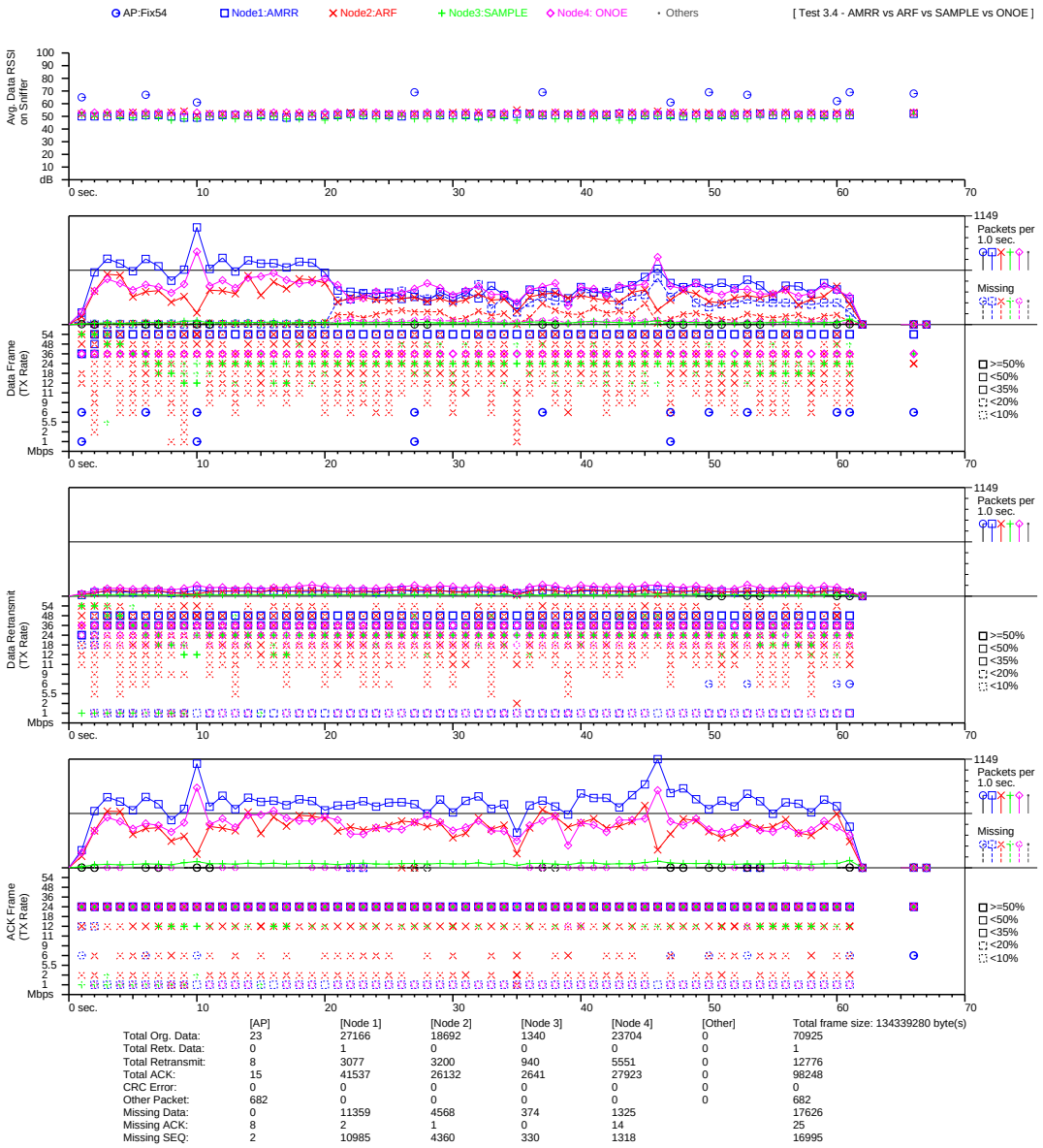


Figure B.36: Test 3.4 - AMRR, ARF, SAMPLE, ONOE (Full Graph)

Appendix C

802.11 Frame Sequence Number Reordering

Refer to IEEE 802.11 standard, there is the Sequence Control for Data and Management Frame. It is a 12-bit number in range of 0 to 4095. It is incremented by one for each new frame to be sent and will wraparound to 0 when it is passed through 4095. The sequence number of retransmission is same as original frame.

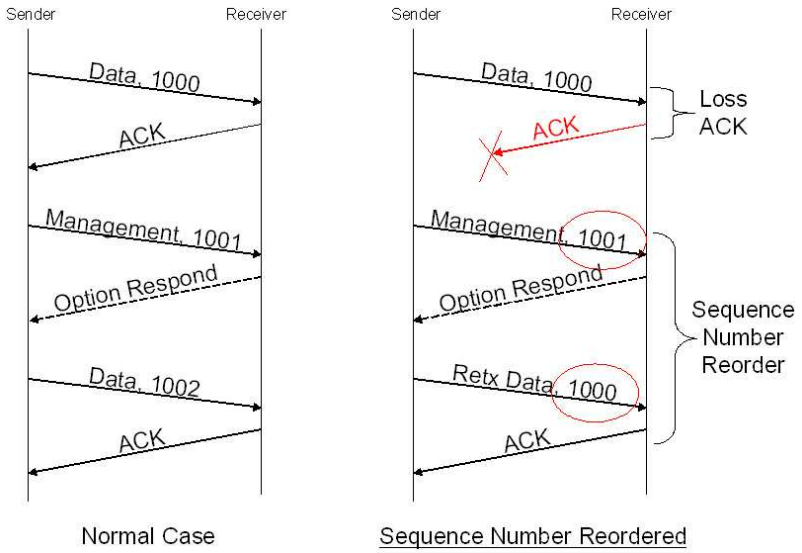


Figure C.1: Case of Sequence Number Reorder

The normal case of Sequence Control is shown in left-hand size of Figure C.1. The sequence number space is shared for Data and Management Frame. In case of some ACK frame is lost, the Sequence Number Reorder will occur, as depicted in the right-hand side of Figure C.1. This may be due to a problem for sequence number related analysis or algorithm. This also discloses that the retransmission mechanism in MadWiFi is independent of the Management Frame output queue in layer 2. It is directly handled by the HAL from Atheros's supplier.