T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NS-2 VE NS-3 AĞ BENZETİM YAZILIMLARININ KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Ünal ÇAVUŞOĞLU

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Ahmet ZENGİN**

Ocak 2014

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NS-2 VE NS-3 AĞ BENZETİM YAZILIMLARININ KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Ünal ÇAVUŞOĞLU

Enstitü Anabilim Dalı	:	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	:	**BİLGİSAYAR MÜHENDİSLİĞİ**

Bu tez 03 / 01 /2014 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Doç. Dr. Ahmet ZENGİN	Doç. Dr. Celal ÇEKEN	Yrd. Doç. Dr. Fatih ÇELİK
Jüri Başkanı	**Üye**	**Üye**

TEŞEKKÜR

Yüksek lisans eğitimim süresince değerli birikimlerini bana aktaran, tezimin başlangıcından bitimine kadar her aşamasında sorunlarımı dinleyen, çalışmalarıma yön veren ve değerli zamanını sorunlarımın çözümüne ayıran tez danışmanım Sayın Doç. Dr. Ahmet ZENGİN' e, tez ile ilgili çalışmam da bilgi ve birikimlerinden yararlandığım değerli hocalarım ve arkadaşlarıma teşekkürlerimi sunarım. Ayrıca maddi ve manevi desteklerini esirgemeyen aileme ve üzerimde emeği olan herkese sonsuz teşekkürler.

İÇİNDEKİLER

BÖLÜM 6.

SİMGELER VE KISALTMALAR LİSTESİ

NS-2	: Network Simulator-2
NS-3	: Network Simulator-3
NAM	: Network Animator
IEEE	: The Institute of Electrical and Electronics Engineers
PDNS	: Parallel/Distributed Network Simulator
GTNetS	: The Georgia Tech Network Simulator
NIC	: Network Interface Card
TCL	: Tool Command Language
OTCL	: Object Tool Command Language
OSI	: The Open Systems Interconnection
DARPA	: Defense Advanced Research Projects Agency
SSFNET	: Scalable Simulation Framework Network
WIMAX	: Worldwide Interoperability for Microwave Access
AODV	: Ad hoc On-Demand Distance Vector
DSDV	: Destination-Sequenced Distance-Vector Routing
DSR	: Dynamic Source Routing
BGP	: Border Gateway Protocol
TCP	: Transmission Control Protocol
UDP	: User Datagram Protocol
LBL	: The Lawrence Berkeley National Laboratory
VINT	: Virtual InterNetwork Testbed Project
JIST/SWANS	: Java in simulation time/scalable wireless ad hoc network sim.
LTE	: Long Term Evolution
GNU/GPLv2	: The GNU General Public License version 2
CSMA	: Carrier Sense Multiple Access
IP	: Internet Protocol
MAC	: Media Access Control Address

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Ağ simülatörleri, NS-2, NS-3, Performans analizi, Karşılaştırma

Bu tezde, günümüzde bilgisayar ağları simülasyonunda kullanılmakta olan ağ simülatörleri üzerine bir çalışma gerçekleştirilmiştir. Bilgisayar ağları simülasyonlarında yaygın olarak kullanılmakta olan yazılımların tanıtımı yapılmış ve programlar hakkında kısaca bilgiler verilmiştir. Ağ simülasyon yazılımlarını, açık ve kapalı kaynak kodlu olarak iki kısma ayırmak mümkündür. Bu çalışma da, açık kaynak kodlu ağ simülasyon yazılımları içerisinde ön plana çıkan ve üzerinde bir çok çalışma gerçekleştirilmiş olan NS-2 ve NS-3 ağ simülatörleri hakkında detaylı bilgiler verilmiş, bu iki simülatörler arasında bir karşılaştırma yapılmıştır.

NS-2 ve NS-3 ağ simülatörleri üzerinde modellenen ağlarda; düğüm sayısı, ağ çıkış değerleri, simülasyon zamanı, işlemci ve bellek kullanım miktarları gibi parametreler üzerinden, farklı donanım özelliklerine sahip bilgisayarlar ve farklı yapıdaki ağlar üzerinde yapılan testler sonucu ağ simülatör programlarının performansları değerlendirilmiştir.

COMPARISON OF THE NS-2 AND NS-3 NETWORK SIMULATOR

SUMMARY

Key Words: Network simulator, NS-2, NS-3, Analysis of performance, Comparison

In this thesis, a study is conducted on the network simulator programs that simulate computer networks for processing. A simulation of computer networks, which is widely used simulation software, is introduced briefly. Network simulation software can be divided into two parts: open and closed source code. In this study, detailed information about open source network simulation software that is about NS-2 and NS-3 network simulators is given. Scalability and performance analysis of these two simulators have been evaluated. Large-scale networks are built on NS-2 and NS-3 network simulators; using different parameters, such as number of nodes, the simulation time, the processor and memory amounts. In tests conducted on computers with different hardware specifications, performances of the network simulator programs are evaluated.

BÖLÜM 1. GİRİŞ

1.1. Giriş

Günümüzün ilerleyen teknolojisinin bir getirisi olarak, bilgisayar ağlarına duyulan ihtiyaç oldukça artmıştır. İnternet üzerinden gerçekleşen iletişimin artması ile dünya üzerinde iletişim çok kolay hale gelmiştir. Bunun bir sonucu olarak bilgisayar ağları üzerinde iletimi gerçekleşen veri miktarı ve yoğunluk her geçen gün artış göstermektedir. Bilgisayar ağları üzerinde yapılacak olan işlemlerin gerçek ortama uygulanmadan önce bilgisayar ortamında uygulanması ve tasarımlarının bu uygulamalar neticesinde gerçekleştirilmesi bilgisayar ağları üzerinde çalışan araştırmacılara çok büyük avantajlar sağlamaktadır. Bilgisayar üzerinde gerçekleşecek her türlü işlemi gerçek ortama ihtiyaç duyulmadan sanal ortamda gerçeklenmesi bu şekilde mümkün olmaktadır. Bilgisayar ağ simülasyon yazılımlarının kullanıldığı bir çok alan mevcuttur. Ticari anlamda bu işi gerçekleştiren şirketlere ait olan yazılımları kullanacak olan kurumlar yüksek miktarlarda ödemeler yapmak zorunda kalmaktadır. Bu tez çalışmasında yaygın olarak kullanılan ağ simülatörleri hakkında bilgilendirmenin ardından, açık kaynak kodlu ağ simülatörleri arasında yaygın bir kullanıma sahip NS-2 ve 2006 yılında geliştirilmeye başlanan NS-3 simülatörleri hakkında bilgiler verilmiş ve NS-2 ve NS-3 ağ simülatör programları üzerinde farklı topolojilerde senaryolar geliştirilerek, simülatörlerin performans analizleri ve karşılaştırması gerçekleştirilmiştir.

Test sonuçlarına göre ağ simülatörlerinin performansları hakkında değerlendirmelerde bulunulmuştur. Farklı donanımsal özelliklere sahip bilgisayarlar üzerinde, ızgara ve yıldız ağ topolojik yapıları üzerinde testler yapılmıştır. Yıldız topolojide 500 düğüm için, ızgara topolojide ise 256 düğümden oluşan bir ağ yapısı üretilmiştir. Simülasyonların tamamlanma süreleri, cpu, bellek kullanım durumları incelenmiştir. Simülasyon sonucu, ağ simülatör programların üretmiş olduğu

animasyon dosyaları Nam ve NetAnim programlarında çalıştırılarak, gerçekleşen simülasyonların izlenmesi sağlanmıştır. Ayrıca ağ simülatörlerinin üretmiş olduğu NS-2'de tr uzantılı dosyalar, tracegraph programında grafiksel olarak üretilmiş ve incelenmiştir. NS-3 ağ simülatörünün üretmiş olduğu, pcap dosyaları wireshark programında çalıştırılarak simülasyon ile ilgili detaylı analiz bilgileri elde edilmiştir.

Yapılan testler sonucunda, her iki topoloji üzerinde, NS-2 simülatöründe simülasyonun tamamlanma süresinin NS-3 simülatörüne göre daha kısa olduğu gözlemlenmiştir. Kaynak kullanımı açısından değerlendirildiğinde ise, simülasyonlarda NS-3 ağ simülatörünün daha verimli bir şekilde kullanım gerçekleştirdiği tespit edilmiştir. NS-3 ağ simülatörünün gelişmiş mimarisi ve üzerinde çalışan büyük bir ekip ve maddi desteğe sahip olması, paralel ve dağıtık yapıdaki programlamayı desteklemesinden ötürü, daha geniş ve kapsamlı sistemlerin simülasyonu için uygun olduğu, yani daha ölçeklenebilir ve performanslı olduğu söylenebilir.

1.2. Yapılan Çalışmalar

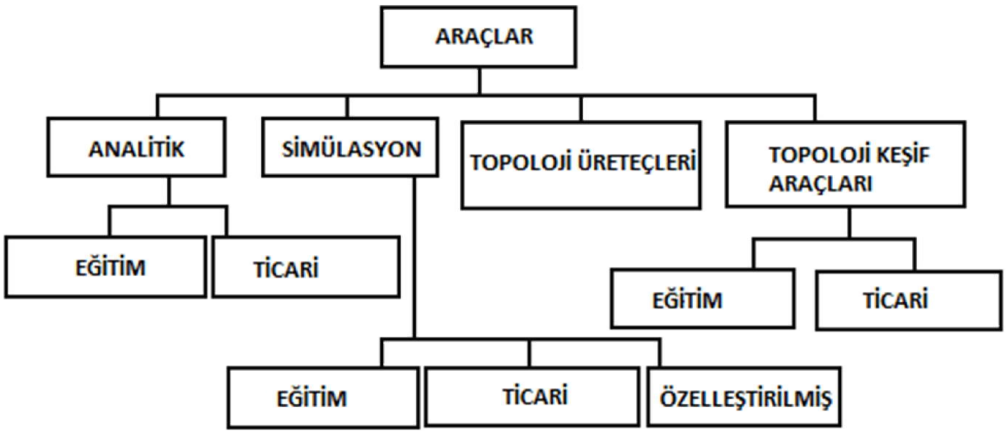
Günümüzde kullanılmakta olan birçok ağ simülatörü vardır. Geliştirilmiş olan ağ simülatörlerinin özellikleri incelendiğinde birçok noktadan karşılaştırma yapmak mümkündür. Bu bölümde literatürde ağ simülasyonlarının karşılaştırlması üzerine gerçekleştirilmiş olan çalışmalardan bazıları incelenecektir.

Siraj ve arkadaşları 2012 yılında ağ simülasyon programlarına ait gerçekleştirdikleri çalışmada, NS-2, NS-3, Opnet, Netsim, OMNeT++, REAL, J-Sim ve QualNet simülatörleri hakkında bilgiler verilmiş ve simülatörler tanıtılmıştır. Bu simülatörlerin avantaj ve dezavantajları açıklanmış, simülatörler üzerinde örnek bir simülasyon senaryosu gerçekleştirilmiş ve sonuçlar karşılaştırılmıştır (Siraj ve ark., 2012).

Rachna ve arkadaşları 2012 yılında gerçekleştirdikleri çalışma da ağ simülatörlerinin kullanılma gerekçelerini açıklamış ve ağ kurulumlarında simülatör kullanımının önemine vurgu yapmışlardır. Ağ simülatörlerinin kullanımı, zamandan ve maliyetten

ciddi anlamda kazanç sağlamaktadır. Bu çalışmada özellikle açık kaynak kodlu ağ simülatör programlarından NS-2 ve NS-3 ağ simülatör programları hakkında kapsamlı bilgiler verilmiş, simülatör mimarileri katmansal bazda açıklanmış, avantaj ve dezavantajlarından bahsedilerek, NS-3 simülatöründe planlanan gelişimler açıklanmıştır. İki simülatör bir çok kriter bakımından kıyaslanarak değerlendirme yapılmıştır (Rachna ve ark., 2012).

Rahman ve arkadaşları 2009 yılında gerçekleştirdikleri çalışmada ağ modelleme ve simülasyon araçlarını sınıflara ayırarak bu sınıflar altında programların incelemesini gerçekleştirmişlerdir. Çalışmada 100'e yakın program incelenmiş ve sınıflandırmaya tabi tutulmuştur. Modelleme ve simülasyon araçları başlıca analitik, simülatör, topoloji üretici ve topoloji keşfi için kullanılanlar olmak üzere dört kısma ayrılmıştır. Bu bölümlerde, daha alt kısımlara ayrılarak detaylı bir incelemeye tabi tutulmuş, birbirine göre avantaj ve dezavantajları konusunda bir değerlendirme gerçekleştirilmiştir. Özellikle simülatörler alt başlığında bulunan ticari ve eğitimsel amaçlı kullanılan ve büyük ölçekli ağların simülasyonu için kullanılan programlar bir çok kritere göre karşılaştırılmıştır. Şekil 1.1' de bu çalışmada gerçekleştirilen sınıflandırmaya ait bir diagram görülmektedir (Rahman ve ark., 2009).



Şekil 1.1. Ağ modelleme araçlarının sınıflandırılması

Zengin gerçekleştirmiş olduğu çalışmada, bir çok ağ simülatörüne ait detaylı bilgiler vererek, ağ simülatörlerini farklı kriterler üzerinde kıyaslayarak değerlendirmelerde bulunmuştur. DEVS-JAVA tabanlı bir ağ simüaltörü üzerinde çalışamalar

gerçekeştirilirek, ağ simülatörünün ölçeklenebilirlik ve performans analizi gerçekleştirilmiştir (Zengin, 2011).

Weingartner ve arkadaşları 2009 yılında yapmış oldukları çalışmalarında (Weingartner ve ark., 2009) NS-2, NS-3 ve OMNeT simülatörlerini karşılaştırmışlardır. Simülasyon gerçekleşme zamanı ve bellek kullanımı açısından performans değerlendirmesi yapmışlardır. Simülasyon gerçekleşme zamanına göre NS-3 ve OMNeT++ birbirine yakın sürelerde, NS-2 ise daha uzun sürede işlemi gerçekleştirmiştir. Bellek kullanımı açısından ise, NS-3' ün belleği en efektif kullandığı, NS-2'nin en yüksek kullanım oranına sahip olduğu tespit edilmiştir.

Becker ve arkadaşları yaptıkları çalışmada (Becker ve ark., 2008) OMNeT++, NS-2 ve OPNET simülatörlerinin performans karşılaştırmasında saniyedeki üretilen paket sayısına göre karşılaştırma yapılmıştır. NS-2 (1.59 paket/s) ve OPNET (1.65 paket/s) den birbirine yakın değerler elde edilmiştir. OMNeT++ (1.19 paket/s) ise diğer programlara göre daha düşük değere sahip olduğu görülmüştür.

Tamara yüksek lisans tez çalışmasında, su altında çalışacak, akustik özellikleri kullanan bir kablosuz algılayıcı ağ simülatörü üzerinde çalışmıştır. Çalışmada ağ simülatörlerinden NS-2, NS-3, OPNET, OMNET++, kablosuz algılayıcı ağların simülasyonunda kullanılmak üzere tasarlanmış olan Sensim, Castalia, Aqua-Sim, Sense ağ simülatörleri hakkında bilgi verilmiştir. Simülatörün tasarımı sırasında OMNET++ ağ simülatörü ile Mixim çerçevesi kullanılmıştır. Java ortamında geliştirilen simülatör arayüzü üzerinden örnek simülasyonlar uygulanarak program test edilmiştir (Tamara, 2011).

Sarkar ve arkadaşları yapmış oldukları çalışmada ağ simülatörünü ticari ve açık kaynak kodlu olmak üzere 2 kısma ayırmışlardır. Bu alanda sık kullanılan ve tanınmış ağ simülatörleri hakkında bilgi vererek, simülatörlerin karşılaştırması yapılmıştır. Çalışmada simülatörlerin destekledikleri protokol yapıları, zayıf yönleri, simülasyon metot ve teknikleri detaylı bir şekilde ortaya konulmuştur. Ayırca IEEE' nin yayınlarında bu ağ simülatörleri ile ilgili 2007-2009 yılları arasında yayınlanmış olan çalışmalar üzerinde, istatistiki bir inceleme yapılmıştır. Yayın sayıları verilerek

karşılaştırmıştır. 2007-2009 yılları arasında IEEE menşeli yayınlanan 8370 yayından büyük bir çoğunluğunun NS-2 ağ simülatörü kullanılarak gerçekleştirildiği tespit edilmiştir (Sarkar,2011).

Literatürde ağ simülatörlerinin karşılaştırılması ile ilgili yapılmış daha bir çok çalışma bulunmaktadır (Nicol, 2003) (Gupta ve ark., 2013) (Varga ve ark., 2008) (Chaudhary ve ark., 2012). Fakat bu konu başlı başına ayrı bir tez konusu olabilecek bir niteliktedir.

1.3. Tezin Amacı

Tezin amacı, bilgisayar ağ simülasyonu hakkında genel bir bilgilendirmenin ardından ağ simülasyonu için kullanılmakta olan birçok ağ simülatörü hakkında kısaca bilgi vererek, açık kaynak kodlu NS-2 ve NS-3 ağ simülatörlerini detaylı olarak tanıtıp, farklı topolojiler üzerinde ve performans testleri gerçekleştirerek değerlendirmelerde bulunmaktır.

1.4. Bilime Katkısı

Bu tezde bilgisayar ağlarının simülasyonun da yaygın olarak kullanılmakta olan açık kaynak kodlu ağ simülatörlerinden NS-2 ve NS-3 ağ simülatörleri hakkında detaylı bilgi verilmiş, karşışlaştırma yapılmış ve performans analizleri gerçekleştirilmiştir. Ağ simülasyonu üzerinde çalışacak araştırmacılar için bu tez çalışmasının önemli bir yol gösterici olacağı kanaatindeyim. Ayrıca çalışma ağ simülatörleri üzerine ciddi bir literatür taraması içermektedir.

1.5. Tez Planı

Tez çalışmasında giriş ve literatür kısmının ardından, bilgisayar ağ simülatörleri hakkında gerekli temel bilgiler verilmiştir. Her bölümde konu ile ilgili yapılmış olan çalışmalar ile ilgili literatür taramaları aktarılmıştır. 2. Bölümde çalışma ile ilgili temel bilgiler verilmiş, 3. Bölümde yaygın olarak kullanılan açık ve kapalı kaynak kodlu ağ simülatörleri hakkında açıklamalar yapılmıştır. 4. Bölümde NS-2 ağ

simülatörü hakkında, 5. Bölümde NS-3 ağ simülatörü hakkında detaylı bilgilerin verilmesinin ardından, 6. Bölümde ise NS-2 ve NS-3 ağ simülatörlerinin performans analizinin gerçekleştirilmesi için, testler yapılmış ve daha önce gerçekleştirilen çalışmalar verilmiştir. Son bölümde ise sonuç ve değerlendirme yapılmış ve gelecekteki çalışmalar hakkında bilgi verilmiştir.

BÖLÜM 2. TEMEL BİLGİLER

2.1. Simülasyon Nedir?

Simülasyon, gerçekte varolan bir sistemin veya sürecinin zamana bağımlı olarak gerçekleştirilmesidir (Banks ve Carson,1984). Sistemi oluşturan nesneler arasında tasarım sırasında belirlenen ilişkileri barındıran bir süreç modelidir. Simülasyon modellemeye yarayan bir araçtır. Simülasyon varolan veya daha sonraki dönemlerde gerçekleştirilebilecek olan işlemler ile ilgili gerçek bilgiler elde edilmesine yardımcı olur. Simülasyon sırasında modellenecek olan yapı bilgisayar vasıtasıyla modellenmektedir.

Simülasyon dinamik bir sistemin özelliklerini ve davranışlarını bilgisayar aracılığıyla değerlendiren bir tekniktir. Kullanıcısına değişik tasarım ve işletim stratejilerinin genel sistem performansı üzerindeki etkisini gösterir. Sonuçta elde edilenler, istenen model karakteristiklerine ait birer tahmindir. Diğer bir tanımla simülasyon, incelenen bir gerçek dünya sisteminin belli bir zaman diliminde istenilen gerçek karakteristiklerini tahmin etmek amacıyla sistemin matematiksel, mantıksal bir modelinin geliştirilmesi ve bu sistem üzerinde deneyler yapılması sürecidir (Cournat, 1943).

Simülasyon deneysel bir metottur. Gerçek bir ortamda deneylerin gerçekleştrilmesinin yerine, testler simülasyon üzerinde yapılır. Simülasyon sistemlerini kullanmanın birçok avantaj ve dezavantajları vardır. Simülasyonları gerçekleştirmek için uygun yazılımlar bulunmaktadır. Ancak maliyetlerinin yüksekliği nedeniyle satın alımları problem olabilmektedir. Bazı grafik tekniklerine dayanan simülasyon yazılımları geliştirilmiştir. Simülasyon programlarının kullanımı sayesinde, simülasyon modellerinin gerçeklenmesi otomatik yapılabilmektedir. Simülasyon sonuçlarının doğruluğu, modelin gerçek sisteme yakınlığı ile doğru

orantılıdır. Gerçeğine daha yakın model daha fazla ayrıntı gerektirir. Buna paralel olarak modelin tasarlanması ve simülasyonun gerçekleştirilmesi daha uzun sürer.

Modelleme ve simülasyon bir sistemin gerçek ortama aktarılmadan önce, sistem üzerinde gerekli test ve deneylerin gerçekleştirilip, çalışılabilirliğinin test edilmesi için kullanılmaktadır. Sistem üzerinde farklı parametrelerin değişmesi sonucu sistemin davranışının gözlemlenmesi noktasında faydalanılmaktadır. Bu değişimler simülasyon üzerinde farklı parametre değerleri ile uygulanabilir ve karşılaştırılabilir.

Modelleme ve simülasyon sistem analistlerinin konu ile ilgili daha geniş ve kapsamlı bir şekilde düzenleme yapmalarına olanak tanır. Analitik olarak oluşturulmuş modellerin gerçeklenmesi, gerçek ortamlarda olayların gözlemlenmesi ve değişim yapılması birçok zorluğu beraberinde getirmektedir. Simülasyon ve modelleme bu gibi zorlukların üstesinden gelinmesinde avantajlar sağlamaktadır. Gerçek ortamda yapılan test ve deneyler çok maliyetli olmakta ve sistem üzerinde değişiklik yapmak imkânsız bir hal almaktadır. Simülasyon kullanımı sayesinde, sistemin üzerinde alternatif yöntemler arasında bir mukayese yapılması, var olan sistemler üzerinde, sisteme zarar verilmeden testlerin gerçekleştirilmesi mümkün olmaktadır.

Simülasyon yöntemleri bir çok avantajının yanında, çok karmaşık sistemlerin oluşturulmasındaki zorluklar, simülasyonun gerçekleştirilmesi için bir çok analitik çalışmanın yapılması için sarf edilen emek ve zaman, simülasyon işlemlerinin yapılmasında bilgisayara bağımlı olması, alternatif bir çok çözüm üzerinde denemelerin yapılması gibi dezavantajları da bulunmaktadır.

Simülasyon ve modelleme işlemleri için kullanılan, birçok alan da geliştirilmiş olan paket yazılımlar ve simülatörler bulunmaktadır. Bazı yazılımlarda görsel bir arayüz olmaksızın sadece kodlama ile simülasyon oluşturulmakta, diğer tarafta ise görsel bir arayüze sahip ve neredeyse hiçbir programlama gerektirmeden simülasyonun gerçeklenmesi sağlanmaktadır. Simülasyon işlemlerinin karmaşıklığını azaltmak için bu türden bir paket programın kullanımı işlemleri kolaylaştıracaktır. Simülasyon sistemlerinin kullanımı, sistemlerin modellenmesinde analitik modellere göre daha başarılı olmaktadır. Güçlü bilgisayarlar üzerinde gerçekleştirilen simülasyonlar da

sistemi anlık olarak takip etmek mümkündür. Sistemin gerçek ortamda uygulanmadan önce ortaya çıkabilecek olası durumlar modellenen simülasyon ortamlarında önceden incelenebilmektedir. Simülasyon çeşitleri ayrık ve sürekli olay olmak üzere iki başlık altında incelenebilir.

2.1.1. Ayrık olay simülasyonu

Ayrık olay simülasyonu, simülasyonun gerçekleşmiş olduğu zaman dilimi içerisinde, gerçekleşecek olayların bir sıralama şeklinde tanımlanıp, gerçekleştirilmesidir. Belirlenen zaman dilimi içerisinde ilgili zaman dilimine ait olan değişken aynı değeri taşımaktadır yani belli zaman dilimleri içerisinde sürekli bir değişim söz konusu değildir. Bu simülasyon tasarımında tasarlanan model içerisinde gerçekleşecek olayların sırasını kontrol etmek mümkündür. Eğer simülasyonun zamanı sonlu bir yapıya sahip ise ayrık simülasyon metodunun kullanılması uygun olacaktır. Bu yapıda kullanıcı tüm çevreyi gözlemleyebilir (Rubinstein, 1993).

Simülasyon sırasında kullanılan algoritma belirlenen artış zamanına göre gerçekleşecek olan ve gerçekleşen olayların durumlarını kontrol eder. Bu değerlendirme işlemi simülasyon yöntemi için bir dezavantajdır. Çünkü her adımda tüm durumların kontrolü ciddi zaman gerektirebilmektedir. Fakat yapı olarak bakıldığında basittir ve yüksek seviyeli diller ile birlikte uygulanması kolaydır. Simülasyon tasarımı gerçekleştirilmesinde işlemler bir blok diagramı olarak tasarlanır ve blokların süreç içerisinde işlem görme zamanları ayarlanmaktadır. Ayrık olay simülasyonu, sayısal veri iletişim sistemleri ve bilgisayar ağları, mesajların üretimi ve dağıtımı gibi durumlar gerçekleştiğinde modüllerin çalıştırıldığı ve simülasyon saatinin ilerlediği ayrık olay simülasyon yöntemi ile modellenebilmektedir. Bu simülasyon türüne basit bir örnek verecek olursak, bankada işlem gerçekleştirmek için beklemekte olan veya işlem gerçekleştirmekte olan müşterilerin zaman içerisindeki durumları incelenebilir. İşlemi gerçekleştirmek için bekleyen müşteri sayısı veya işlem gerçekleştiren müşteri sayısı belli bir zaman dilimi içerisinde sabit bir değer almaktadır.

2.1.2. Sürekli olay simülasyonu

Sürekli olay simülasyonunda, simülasyonun gerçekleştiği süre içerisinde olaylar ve ilgili parametreler sürekli bir değişim gösterebilmektedir. Değişkenler her hangi bir zamanda diğer bir parametreyide etkileyecek şekilde değişebilmektedirler. Sistemin yapısından dolayı gerçekleşen olaylar zamana bağlı olarak sürekli bir değişim içerisindedirler (Pollatschek,1996). Simülasyon sırasında gerçekleşecek sonsuz sayıda olay olacağı öngörülmektedir.

Sürekli olay simülasyonu için basit bir örnek ise, bir barajdaki su seviyesinin zamana bağlı olarak değişimi olabilir. Barajdaki su miktarı zaman içerisinde birçok parametre tarafından etkilendiği için sürekli bir değişim göstermektedir. Baraja dahil olan ve barajdan çıkış yapan su sürekli farklılık göstermektedir ve farklı zamanlarda çok farklı değerler alabilmektedir. Bu örnekte sürekli simülasyon yönteminin kullanımı oldukça uygundur. Ayrık simülasyon yöntemide kullanılarak bir modelleme gerçekleştirilebilir fakat başarılı bir simülasyon gerçekleştirilemez.

2.2. Ölçeklenebilirlik

Ölçeklenebilirlik, donanımın veya yazılımın bilgi işlem gereksinimlerini karşılamak üzere kolayca genişletilebilme yeteneğidir (Daniel, 2012). Her ağ topolojisi veya bu topolojiler üzerinde kullanılan cihazlar, analiz programları, daha iyi performans ve ölçeklenebilirlik sağlayacak şekilde tasarlanmalıdır. Bununla birlikte, donanım yapılandırmaları ve gerektiği gibi iyi tasarlanmamış veya gerektiği gibi sınanmamış uygulamalardaki performans sorunları, ölçeklenebilirliği olumsuz etkileyebilir. Sistem analistleri, donanımı, dağıtılmış uygulamaları ve özel bileşenleri de kapsayan altyapılarını dikkatle planlayarak ve değerlendirerek, ağ yapılarının ölçeklenebilirliğini önemli ölçüde arttırabilirler. Günümüz teknolojisinin çok hızlı gelişmesi sonucunda kurulacak olan sistemlerin veya tasarlanacak programların ölçeklenebilir olması çok önemli bir ölçüt haline gelmiştir.

Jonathan B. Harris 2005 yılında yapmış olduğu doktora tezinde bilgisayar ağlarının ölçeklenebilirliği ve genişletilebilmesi üzerinde bir çalışma gerçekleştirmiştir. Birebir anlık güncelleme gerektiren uygulamaların artması ile bu sistemlerin üzerinde çalışan bilgisayar ağlarıda ciddi anlamda önem kazanmıştır. Sistem kaynaklarını oldukça fazla tüketen bu uygulamaların üzerinde çalıştığı ağ yapılarının da olabildiğince efektif bir şekilde dizayn edilmesi gereklidir. Bununla birlikte gittikçe büyüyen sistemler üzerinde yapılan simülasyon çalışmaların gerçeğe yakınlığı çok önemlidir. Yapılan çalışmada peer-to-peer ağlar üzerinde gerçekleştirilen simülasyonların ölçeklenebilirlik ve genişletilmesi üzerine testler gerçekleştirilmiştir. Ölçeklenebilir bir simülasyon ortamı hazırlanmış ve gnutella protokolü kullanılmıştır. Farklı protokol yapılarının simülasyon performansı testedilmiştir (Jonathan, 2005).

2.3. Bilgisayar Ağ Simülasyon Programlarının Genel Özellikleri

Simülasyonun kullanıldığı birçok farklı alan mevcuttur. Savunma, eğitim, tıp, bilişim ve daha çok bir alanda, alana özgü tasarlanmış ve kullanılan simülasyon uygulamaları bulunmaktadır. Ağ simülatörleri aşağıda belirtilen işlemleri gerçekleştirmek için kullanılabilir:

1. Haberleşme trafiğinin modellenmesi,
2. İletişim kuralı modelleme,
3. Ağ modellemesi,
4. Çok işlemcili ve diğer dağıtık donanım sistemlerini modelleme,
5. Donanım yapılarını inceleme,
6. Karmaşık sistemlerin performans durumlarının değerlendirilmesi ve
7. Ayrık olay yaklaşımının elverişli olduğu diğer sistemlerin modellemesi.

Bu bölümde bilgisayar ağlarının simülasyonunda kullanılan bilgisayar ağ simülasyon programlarının taşıdığı genel özellikler ve kullanım alanları anlatılacaktır. Bilgisayar ağları üzerinde, simülasyon işlemlerini gerçekleştirmek üzere bir çok simülatör bulunmaktadır. Bu simülatörlerin ağların yapısına uygun olarak işlem yapılabilmesi

için bazı özellikleri taşıması gerekmektedir. Burada özellikle NS-2 ve NS-3 ağ simülatörlerinin taşımış olduğu özellikler üzerinden açıklamalarda bulunulacaktır.

Model genişlenebilirliği: Çoğu araştırmacı, kullanıcıları yeni simülasyon scriptleri yazarak simülatörü genişletmek, değişiklik yapmak ya da yeni modeller yazmak isterler. Model değişikliğini kolaylaştırmak için, NS-3 polimorfik sınıflarla, nesne tabanlı tasarımı kullanmaya devam etmektedir. İzin verilen kullanıcılar değiştirmek istediği açıları istediği şekilde değiştirebilir. Yeni modellerin eklenmesini kolaylaştırmak için NS-3, derleme zamanı için bileşen tabanlı mimariyi ya da yeni modellerin çalışma zamanı eklentisini kullanır.

Simülasyon kodunun tekrar kullanımı: Çoğu kullanıcı var olan kodu adapte ederek, çalışmalarını NS-2 ile gerçekleştirirler. Bazı genel kodlar, alt sınıf nesnelerinin çalışma zamanı yerleşimi için izin verilen temel sınıf nesne pointerlarına göre yazılır. NS-3 simülasyon kodlarının tekrar kullanımını kolaylaştırmak için bazı teknikler kullanmaktadır. Bunlardan bazıları, var olan sınıfları genişletmek için kalıtım, topoloji nesnelerinin hazırlığı, kolaylıkla değiştirebilir simülasyon çerçevesi, örnek script deposu ve çalışma zamanı konfigurasyon sınıfları ve varsayılan değerler için bir sistemdir.

Çalışma zamanı konfigurasyonu: NS-3, kullanıcıların varsayılan değerleri tekrar tanımlamaları için izin verilen esnek bir teknik ve simülatörün tekrar derlemeyen sınıf tiplerini sağlamaktadır. Varsayılan değer veritabanı komut satırı arguman ayrıştırımı kolaylığı ile entegre edilir.

Ölçeklenebilirlik: NS-3, simülasyonların ölçeklenebilirliğini geliştirmek için teknikleri içerir. Bunlardan bazıları, PDNS ve GTNetS ile tanıtılan dağıtık simülasyon teknikleri, hesaba dayalı yoğun sonuçların ön belleğe alınması ve kablosuz simülasyon için tanıtılan ölçeklenebilirlik teknikleridir.

Yazılım entegrasyonu: NS-3 yönlendirme programları, uygulamalar ve kernel kodu gibi var olan yazılımların yeniden kullanımına yönelik bir simülatör programıdır. Tasarım, kapsülleme teknikleri üzerine inşa edilir. Bu teknikler uygulamadan

arayüzü ayrıştırır. Gerçek dünyadaki aygıtların mimarisini oluşturur ve her iki gerçek ve simüle edilmiş ortamlarda çalışacak uygulama kodlarına izin veren soyut bir kütüphane içerir. Simülasyon ve aynı zamanda PlanetLab, Emulab ve ORBIT gibi olanaklarla deneysel bileşen içeren ağ araştırmaları artan bir şekilde devam etmektedir. Araştırmacılar, simülasyon ve deneysel etki alanları arasında daha kolay bir şekilde hareket edebilmeyi istemektedirler. NS-3 dizaynı, simülasyon ve deneyler arasındaki bu etkiyi kolaylaştırmayı amaçlar. Kapsülleme teknikleri simülatör üzerinde kernel kodu ve gerçek uygulamayı çalıştırmaya izin verir.

Senaryo oluşturma: NS-3 kullanıcı arayüzü C++ ana programı üzerine kuruludur ve C++ dili çoğu kullanıcı için tercih edilen bir dil olmaya devam etmektedir. Bununla birlikte, NS-3, kullanıcıların script tanımlamaları ve Pyhton'daki değiştirilebilir bileşenler için pyhton komutları kullanmasına da izin verecektir.

2.4. Bilgisayar Ağ Simülasyonunda Temel Kavramlar

Bu bölümde, bilgisayar ağlarında çok sık kullanılan fakat NS'de özel bir anlama sahip olan bazı terimler açıklanacaktır.

2.4.1. Düğüm

Bir ağa bağlanan bir bilgisayar cihazına terminal veya bazen de uç sistem denir. NS-3 özellikle bir internet simülatörü değil, bir ağ simülatörüdür. NS-3'de temel bilgisayar cihazına düğüm denir. Bu terim C++'da Düğüm sınıfı olarak tanımlanmıştır. Düğüm sınıfı, simülasyonlarda hesaplama cihazlarının gösterimlerini düzenlemek için yöntemler sağlar. Bir düğümü işlevsellik ekleyeceğiniz bir bilgisayar olarak düşünebilirsiniz. Bir kimse bilgisayarın yararlı işler yapmasına olanak vermek için, bilgisayara ilişkili sürücüleriyle uygulamalar, protokol yığınları ve çevre birimi kartları gibi yapıları ekleyebilir.

2.4.2. Uygulama

Genel anlamda, bilgisayar yazılımı iki ana sınıfa ayrılmıştır. Sistem yazılımı bazı hesaplama modellerine göre, bellek, işlemci devreleri, disk, ağ, vb. çeşitli bilgisayar kaynaklarını organize eder. Sistem yazılımı bu kaynakları genellikle doğrudan doğruya kullanıcıya yarayacak görevleri tamamlamak için kullanmaz. Bir kullanıcı, bazı amaçları yerine getirmek için, sistem yazılımı tarafından kontrol edilen kaynaklara sahip olan ve bunları kullanan bir uygulamayı çalıştırabilir. Genellikle, sistem ve uygulama yazılımı arasındaki ayırma satırı, işletim sisteminde meydana gelen öncelik katmanı değişiminde yapılır. NS-3'de gerçek bir işletim sistemi kavramı ve özellikle öncelik katmanları veya sistem çağrıları kavramları yoktur.

NS-3'de simüle edilecek bazı eylemleri oluşturan bir kullanıcı programı için kullanılacak temel terim, uygulamadır. Bu terim C++'da sınıf uygulaması olarak tanımlanmıştır. Uygulama sınıfı, simülasyonlar daki kullanıcı düzeyi uygulamaları düzenlemek için yöntemler sağlar. NS-3 geliştiricilerinden, yeni uygulamalar oluşturabilmek için, uygulama sınıfını nesneye yönelik programlama doğrultusunda özelleştirmeleri beklenir.

2.4.3. Kanal

Ağlarda verinin aktığı ortama kanal denir. Ethernet kablonuzu duvardaki girişe taktığınızda, bilgisayarınızı bir Ethernet iletişim kanalına bağlamış oluruz. NS-3'ün simülasyon ortamında, bir düğüm, bir iletişim kanalını gösteren nesneye bağlanır. Burada temel iletişim alt ağ terimi kanal olarak adlandırılır ve C++'da kanal sınıfı olarak tanımlanır. Kanal sınıfı iletişim alt ağ nesnelerini yönetmek ve bunlara düğümleri bağlamak için yöntemler sağlar. Ayrıca kanallar NS-3 geliştiricileri tarafından, nesneye yönelik programlama doğrultusunda da özelleştirilebilir. Bir kanal özelleştirmesi, bir kablo kadar basit bir şeyi bile modelleyebilir. Ayrıca özelleştirilmiş kanal, büyük bir ethernet anahtarı gibi şeyleri modelleyebilir.

2.4.4. Ağ cihazı

Bir ağa bir bilgisayarla bağlanmak istediğiniz durumlarda özel bir ağ kablosu ve bir donanım cihazı gereklidir. Eğer bu çevre birimi bazı ağ görevlerini yerine getiriyorsa, o zaman ağ arayüz kartı (NIC) olarak isimlendirilir. Günümüzde bilgisayarların çoğu bu ağ arayüz donanımıyla birlikte yapıldıklarından kullanıcılar bu yapı birimlerini görmezler. Bir NIC donanım kontrolünü sağlayan sürücüsü olmadan çalışmaz. Unix'de veya Linux'de bir çevrebirim donanım parçası bir cihaz olarak sınıflandırılır. Cihazlar, cihaz sürücüleri kullanılarak kontrol edilir ve ağ cihazlarıda ağ cihaz sürücüleri kullanılarak kontrol edilir. Unix ve Linux'de bu ağ cihazlarına eth0 gibi isimler verilmektedir. NS-3'te ağ cihaz terimi yazılım sürücülerinin ve simüle edilmiş donanımların her ikisini de kapsar. Simülasyonda bir düğüme yüklenmiş bir ağ cihazı, o düğümün kanallar aracılığıyla diğer düğümlerle iletişim kurması içindir. Gerçek bir bilgisayarda olduğu gibi bir düğüm çoklu NetDevices ile bir kanaldan daha fazlasına bağlanabilir. C++'da ağ cihazı kavramı NetDevice sınıfları tarafından temsil edilmektedir. NetDevice sınıfı, düğüm ve kanal bağlantılarını yönetmek için çeşitli yöntemler sağlar ve nesne yönelimli programcıları tarafından özelleştirilebilir.

2.4.5. Topoloji yardımcıları

Gerçek bir ağda, NIC eklenmiş veya yerleşik NIC'li host bilgisayarları bulunur. NS-3'te ağ cihazı bağlanmış düğümler bulunduğunu söyleyebiliriz. Simule edilmiş geniş bir ağda, düğümler, ağ cihazları ve kanallar arasında birçok bağlantıyı düzenlemeniz gerekir.

Ağ cihazlarını düğümlere ve kanallara bağlamak, IP adreslerini atamak, vb. işlemler NS-3'te genel görevlerden olduğu için, bunu olabildiğince kolay yapabilmek için topoloji yardımcılarını kullanırız. Örneğin, bir ağ cihazını oluşturmak, bir MAC adresi eklemek, bu ağ cihazını bir düğüme yüklemek, düğümün protokol yığınını yapılandırmak ve sonra da ağ cihazını bir kanala bağlamak için birçok farklı NS-3 çekirdek işlemlerini içermektedir.

BÖLÜM 3. AĞ SİMÜLATÖRLERİ

Bu bölümde, yaygın olarak kullanılmakta olan ağ simülasyon yazılımlarından bazıları hakkında bilgi verilecektir.

3.1. OPNET

OPNET, bilgisayar ağları ve iletişim protokolleri için simülasyon, modelleme ve performans analizi işlemlerini gerçekleştirebilen bir ağ simülatör programıdır (Muslim ve ark., 2001). OPNET üzerinde modellenen sistemler ayrık zamanlı simülasyon gerçekleştirilerek analiz edilebilir. OPNET yapısında bulunan modeller hiyerarşik bir yapıdadır. OPNET ağ simülatörünün grafik arayüzleri bulunmaktadır. OPNET ağ simülatörü, ayrıntılı modeller geliştirebilmek için esnek bir yapıda geliştirilmiştir. OPNET, nesne tabanlı olarak tasarlanmış ve geliştirilmiş bir yapıdadır. Yazılım içerisinde yer alan sistemler, nesnelerden oluşmaktadır. Nesnelerinde her birinin kendine göre düzeltme gerçekleştirilecek özellikleri mevcuttur. OPNET simülasyon sonucu olarak çok çeşitli istatistikleri ve grafiksel ifadeleri otomatik olarak üretmektedir. Bu sonuçlar yorumlanarak simülasyon hakkında analiz ve performans değerlendirmesi yapılabilir.

Simülasyon sonucunda elde edilen sonuçlar program ile bütünleşik bir yapıdadır. Ağ simülatörü üzerinde gerçekleşecek simülasyonun kodlarının yazımı sırasında aktif olarak hata ayıklama özelliği bulunan bir ortam sunmaktadır (URL1). OPNET zengin özellikleri barındırmasından dolayı bir çok simülasyon ve modelleme çalışmasında kullanılmaktadır. Yerel alan, geniş alan, kablosuz ağ, mobil ağ uygulamaları, iletişim mimari ve protokolleri gibi birçok alanda uygulamalar gerçekleştirilebilmektedir (URL2).

Literatürde OPNET ağ simülatörü kullanarak yapılan birçok çalışma bulunamaktadır. Xinjie Chang yapmış olduğu çalışmada, bazı ağ simülatör programlarını tanıtmış ve

karşılaştırmalar yapmıştır. Ayrıca OPNET ağ simülatörünü de detaylı bir şekilde tanıtarak örnek simülasyonlar gerçekleştirmiştir (Xinjie Chang, 1999). Hasan ve arkadaşları gerçekleştirdikleri çalışmada mobil ad-hoc network üzerinde dağıtık ağ kontrol sistemlerinin kullanımı modellenmiştir. Simülasyonda, Opnet simülatörü üzerinde DSR ve AODV ad-hoc yönlendirme protokolü, bir çok rasgele olarak oluşturulmuş düğüm ve kontrol noktası kullanılmıştır (Hasan ve ark., 2007).

Hatice DEVELİ yapmış olduğu tez çalışmasında, bilgisayar ağ simülasyonları hakkında verilen bilgilerin ardından Opnet ağ simülatörünü detaylı bir şekilde tanıtmış, Süleyman Demirel Üniversitesi ağ alt yapısını Opnet simülatörü ile modellemiş ve trafik çıktıları elde ederek analizler gerçekleştirmiştir (Develi, 2009). Literatürde OPNET ağ simülatörü kullanılarak gerçekleştirilen birçok çalışma mevcuttur (Zhao, 2011) (Fazeli, 2011).

3.2. OMNET ++

OMNeT++ nesneye-yönelik modüllerin bir araya gelmesi ile oluşmuş olan, bir ayrık olay ağ benzeticisidir. Simülasyonu gerçekleştirilen nesneler, ayrık zamanlarda bir biri ile mesaj gönderimi gerçekleştirerek haberleşme sağlanmaktadır. OMNeT++ ağ simülatörü, C++ ve Tcl/Tk dilleri ile yazılmıştır. En avantajlı olduğu noktalardan biri platform bağımsız çalışmasıdır. Kod üzerinde herhangi bir değişim gerçekleştirmeden farklı ortamlarda çalıştırılabilmektedir. Program ayrıca görsel grafik ve hata ayıklama mekanizmalarına sahiptir. Simülasyon çıktılarının analizi için vektörel ve sayısal değerlerin kayıt edilmesi imkânına sahiptir. Simülatör üzerinde gerçekleştirilecek olan simülasyon işlemleri paralel hale getirilebilir, paralel dağıtık çalışmayı desteklemektedir. Nesnesel yapıda oluşturulan yapılar, tekil veya çoğul modüller tarafından sembolize edilir. Tek simülasyona ait farklı uygulamalar birbirinden farklı parametreler ile uygulanabilmektedir. Simülasyon sırasında oluşturulmuş olan nesneler statik veya dinamik olarak değiştirilebilir (Sırma, 2006).

OMNeT++ modelini, kapsüllenmiş modüller meydana getirmektedir. Katmanlar arası mesaj gönderiminde karmaşık veri yapıları kullanılmaktadır. Modüllerin kendilerine ait parametreleri bulunmaktadır. Her modül üzerinde programlama

gerçekleştirmek için bu parametreler kullanılmaktadır. En alt düzeyde bulunan modüller, o modele ait davranışı belirlemektedir. Gelişmiş kullanıcı ara yüzleri simülatörün kullanımında, simülasyonun kontrolünde ve model içerisinde bulunan nesneler üzerinde kullanıcı tarafından değişiklik gerçekleştirilmesi sırasında kullanıcıya büyük kolaylıklar sağlamaktadır (URL3).

OMNeT++ ağ simülatörü ile gerçekleştirilen literatürde birçok çalışma vardır. Witold ve arkadaşları 2003 yılında OMNeT++ ağ simülatör programını anlatan bir çalışma yayınlamış ve bu çalışmada OMNeT++ simülatörünün mobil uygulamarda kullanımı için bir alt yapı oluşturulmuştur. Mobil uygulamalarda nasıl simülasyon gerçekleştirilebileceği örnekleyerek anlatılmıştır (Witold ve ark., 2003). Micheal 2006 yılında gerçekleştirdiği tez çalışmasında, OMNeT++ simülatörü kullanarak, kablosuz algılayıcı ağlar üzerinde hareketli nesnelerin konumunu belirleme üzerine bir çalışma gerçekleştirmişlerdir (Meer, 2006). Varga ve arkadaşları 2008 yılında yayınladıkları makalede OMNeT++ ağ simülatörü üzerindeki çalışmaları sonucu ortaya çıkan çalışmalarla ilgili bilgiler vermiş, simülatörü tanıtmıştır. OMNeT++ ağ simülatörünün dizaynı, model yapısı, NED dili ile tasarım yapısı, grafiksel yapı modülleri, simülasyon kütüphanelerinin yapısı, paralel simülasyon desteği ve diğer simülatörleri karşılaştırılması yapılmıştır (Varga, 2008).

3.3. J-SIM

J-Sim, Java yazılım dilinde hazırlanmış sınıflardan üretilen nesneleri kullanan bir simulasyon ortamıdır. Kullanımı kolaydır ve önceden hazırlanmış kablosuz algılayıcı ağı paketleri bulunmaktadır. Mimari olarak bileşen tabanlı bir yapıya sahiptir. Bileşen tabanı, simülasyona eklenen bileşenlerin birbirinden bağımsız çalışmasını sağlayabildiğinden, gerçek dünyaya en yakın simülasyon ortamı sağlanmış olmaktadır.

J-Sim, Tcl/Java ve Bağımsız Bileşen Mimarisine dayalı bir simülasyon ortamıdır. J-Sim, Tcl/Java tabanlı olduğu icin Java dilinde hazırlanmış sınıflardan oluşan bileşenleri rahatlıkla kullanmaktadır (URL4, URL5). J-Sim'in uzerine kurulu olduğu mimaride, oluşturulan bileşenler birbirleriyle portları aracılığı ile veri transferi

yaparlar. Birbirlerinden bağımsız hareket edebilen bu bileşenler Java'nın iş parcacıklı yapısı kullanılarak geliştirilmişlerdir. Bu yapı entegre devre tasarımını taklit etmektedir (Sobeih ve ark., 2005). Tcl, J-Sim'inde kullandığı bir script dilidir. Tcl, Java programlama dilinde hazırlanmış paketlere J-Sim ortamından rahatlıkla ulaşabilir. Paketlerin simülasyon ortamına aktarılmasından sonra, paketlerin icinde bulunan sınıflar ve bu sınıfların metotları etkin biçimde kullanılabilir. Tcl, J-Sim'de bileşenlerin gercek zamanlı olarak kullanılabilmesini sağlar. Simülasyonun çalıştığı zamanda bile, bileşenlerin metotları elle değiştirilebilir ve değişkenleri Tcl kullanılarak güncellenebilir. Tcl'in sağladığı kolaylıklardan biri, oluşturulan nesnelere UNIX'de olduğu gibi dosya dizini yapısı ile ulaşılabilmesidir.

Literatürde J-Sim ağ simülatörü kullanılarak yapılmış olan bir çok makale ve tez çalışması bulunmaktadır (Codl ve ark., 2003) (Arnold ve ark., 1996) (Hung-Ying, 2002). J-Sim ağ simülatörüne ait dokümanlara ise resmi web sitesi üzerinden erişim mümkündür (URL6).

3.4. GLoMoSim

GLoMoSim (Global Mobile Information systems simulation library) kablosuz ve kablolu ağ sistemleri için ölçeklenebilir bir simülasyon ortamıdır. Simülasyonların gerçekleşmesi sırasında parsec denilen C tabanlı paralel ayrık olay simülasyonu derleme ortamını kullanır. Glomosim ağ simülatörünün akademik ve ticari sürümleri mevcuttur. Ticari kullanım için glomosim ağ simülastörünün ticari versiyonu olan QualNet kullanılmaktadır. GLoMoSim Kaliforniya Üniversitesinde Mobile Systems Laboratory tarafından geliştirilmiştir. Daha çok kablosuz simülasyon ortamları üzerinde kullanım için tercih edilmektedir. GLoMoSim ağ simülatörü üzerindeki çalışamalar ve güncellemeler 2000 yılı itibariyle durdurulmuştur. Bunun yerine QualNet adı verilen bir ağ simülatörü üzerinde çalışmalar gerçekleştirilmektedir.

Tablo 3.1. Glomosim katmansal yapı ve protokoller

KATMANLAR	PROTOKOLLER
Mobil	Rasgele yol seçimi, iz tabanlı protokoller
Radyo yayılım	İki kanal ve boşluk
Radyo model	Gürültü toplama
Paket alım	SNR, BPSK/QPSK tabanlı BER
Veri bağı	CSMA, IEEE 802.11 ve MACA
Ağ	IP, DSR, LAR, ODMRP, WRP
Taşıma	TCP ve UDP
Uygulama	CBR, FTP, HTTP ve TELNET

Glomosim simülasyonlarında standart OSI referans modeli katmansal yapısı kullanılmaktadır. Tablo 3.1' de katmansal yapı ve bu katmanlarda görev yapan protokoller görülmektedir.

Glomosim ağ simülatörüne ait daha detaylı bilgi Glomosim tanıtım dökümanından elde edilebilir (Jorge, 2004). Literatürde Glomosim ağ simülatörü ile ilgili yapılan bir çok çalışma bulunmaktadır (Lokesh ve ark., 1999) (Zeng ve ark., 1998).

3.5. DEVS - Suite

DEVS-Suite, DEVS yaklaşımına dayalı, açık kaynak kodlu, ayrık olaylı genel amaçlı bir simülasyon ortamıdır (Kim ve ark., 2009). DEVS Suite, paralel DEVS tabanlıdır ve simülasyon sonuçlarının daha iyi izlenebilmesi için bazı eklentiler içeren DEVSJAVA simülasyon aracının yeni bir sürümüdür. DEVS (Ayrık olaylı sistem tanımlama- **D**iscrete **EV**ent **S**ystem **S**pecification) yaklaşımı, ayrık olaylı sistemlerin (DES) modellenmesi ve analizi için kullanılmaktadır. DEVS sistemi, atomik DEVS ve birleşik DEVS olmak üzere davranışları iki farklı seviyede tanımlar. En düşük seviyede, sıralı durumlar arasındaki geçişler gibi ayrık olaylı sistemin otonom davranışını, harici bir girişe nasıl tepki verdiğini ve çıkışı nasıl hesapladığını tanımlar. Birleşik DEVS, daha yüksek bir düzeyde, bir sistemi bileşenler ağı olarak tanımlar. Birleşik DEVS, başka birleşik DEVS bileşenlerine sahip olabildiği için hiyerarşik modelleme yapısı desteklenir. Bileşenler, atomik DEVS modelleri ve

birleşik DEVS modelleri olabilirler. Bağlantılar, bileşenlerin birbirini nasıl etkilediğini gösterir. Özellikle, bir bileşenin çıkış olayları ağ bağlantısı aracılığıyla bir diğer bileşenin giriş olayları olabilir (Zeigler ve ark., 2000) (Zengin ve. Ark., 2009).

3.6. JIST / SWANS

JIST/SWANS ağ simülatörü geniş ölçekli bir kablosuz ağ simülatörüdür. SWANS kablosuz ağ yapılandırmaları ve sensör ağların simule edilmesi için birbirinden bağımsız yazılımsal modüllerin bir araya gelmesi ile oluşmaktadır. NS-2, GloMoSim gibi ağ simülatörlerine benzerlik göstermekle birlikte, daha büyük kapsamdaki ağların simüle edilmesi için de kullanılabilir. Ağ simülatörü geniş kapsamlı ağlar üzerinde simülasyon işlemlerini gerçekleştirirken, performans olarak diğer simülatörlere göre daha yüksek performans sergilemektedir (URL7). JIST standart bir java sanal makinesi üzerinde çalışan bir yüksek performanslı ayrık olay simülasyon motorudur. Bu sanal makine üzerinde simülasyon ortamı oluşturulmakta ve oldukça başarılı sonuçların ortaya çıkmasını sağlamaktadır. Özellikle bellek kullanımı noktasında oldukça verimlidir. Yazılan ve oluşturulan simülasyon kodları java sanal makinesi sayesinde her platformda çalışabilir hale gelmektedir. JIST ağ simülatörüne ait sistem mimarisi incelendiğinde, öncelikle hazırlanan java kaynak kodu derlenir, daha sonra dinamik olarak yeniden yazıcı tarafından modifiye edilir ve sanal makine üzerinde çalıştırılır (Rimon, 2004). JIST ağ simülatörünün resmi web sitesinden yer alan, JIST ve diğer simülatörler üzerinde gerçekleşen simülasyon sonuçları karşılaştırmasına göre JIST ağ simülatörünün diğer ağ simülatörlerine göre daha iyi ağ çıkış değerlerine ve bellek kullanımına sahip olduğu görülmektedir. Literatürde JIST/SWANS ağ simülatörü ile ilgili birçok çalışma bulunmaktadır (Rimon, 2003) (Rimon, 2004).

3.7. NETSIM

NETSIM, Tetcos firması ve Hint Bilim Enstütüsü tarafından ortak geliştirilmiştir. NETSIM ağ laboratuvarlarında deney ve araştırma için kullanılabilen kapsamlı bir ayrık olay ağ simülatörüdür. Ağ simülatörünün ilk sürümü 2002 yılında piyasaya

sürülmüştür. Programın akademik ve ticari versiyonları bulunmakta, bu versiyonlar arasında modül farklılıkları bulunmaktadır. Dünya üzerinde akademik versiyonu 250 adet üniversitede bilgisayar ağı eğitimleri için kullanılmaktadır. NETSIM model kütüphanesi kullanılarak, birçok protokolün uygulaması mümkündür. Ayrıca analiz araçları sayesinde detaylı olarak paket izleme ve performans ölçümlerinin gerçekleşmesine olanak tanır. NETSIM protokol kütüphaneleri ve ağ simülasyonu çekirdek birimi arasında bulunan hizmet geliştirme ortamı kullanıcılara istenilen simülasyonların gerçekleştirilmesi için fırsat sunmaktadır. Protokol kütüphaneleri C dilinde kodlanmış ve açık kaynak kod olarak sunulmaktadır. Kullanıcı bu sayede istediği simülasyonun kodunu oluşturup, protokoller üzerinde değişimler gerçekleştirir. Simülasyonu adım adım ilerleterek, bu sayede işlemleri takip edebilmektedir. Ağ simülatörü ile ilgili daha detaylı bilgiye resmi web sitesi üzerinde akademik ve ticari sürüm özellikleri ve karşılaştırılması, ürün broşürleri, lisans bilgileri ve çalışmalar yer almaktadır (URL8) (Rakesh, 2012). Netsim ağ simülatörüne ait bileşen yapısı resmi web sitesinde bulunmaktadır (URL9). Kullanıcı ihtiyaç duyduğu bileşenleri paketine dahil ederek, sadece kullanılacak olan bileşenleri satın alabilmektedir. Bu şekilde ticari sürüm için maliyetin azaltılması mümkün olmaktadır.

3.8. SSFNET

SSFNet (Scalable simulation framework network models) büyük ve geniş ağ ortamlarının simüle edilmesi için C++ ve Java dillerinde hazırlanmış ayrık olay tabanlı çalışan bir ağ simülatörüdür (Cowie, 1999). SSFNet ağ simülasyon aracı açık kaynak kodlu ve çeşitli ağ simülasyon uygulamalarını barındıran bir yazılımdır. Internet ortamı gibi geniş ölçekli ağ ortamlarının simülasyonları gerçekleştirilebilir veya topoloji istenilen şekilde genişletilebilir. Yalnız genel kullanıcılar için ortamda simülasyon geliştirmek, extra bir dizayn ve analiz aracı sağlamadığı için çok kolay değildir. Ağ modellemesi ve sonuçların analizi bizzat kullanıcının kendisi tarafından yapılmaktadır. Kullanılan nesne tabanlı model ve genişletilmiş framework yapısı sayesinde gerçek ortamdaki yapı çok daha kolay ve gerçekçi bir şekilde simülasyon ortamına aktarılabilmektedir. SSFNet tabanlı bir ağ uygulaması SSF ağ modeli ve ek bileşenlerden oluşmaktadır. Ağ modelleme süreci simülasyon analizi kullanıcıya

göre ve özel ağ tasarımları için çok farklı metotları bulunmaktadır (Yoon ve ark., 2009). SSFNet ağ simülatörü paralel ve dağıtık ağ simülasyonlarına da olanak sağlamaktadır. Ağ simülatörünün ön plana çıkan özelliklerinin başında paylaşımlı bellek mimarisi desteği ve çok işlemcili sistemler üzerinde iyi performanslar ortaya koymasıdır. Ağ simülatörü sistemi içerisinde ana yapıdan türetilmiş iki adet sistem bulunmaktadır. SSF.OS sunucu modellemesi ve protokollerin tasarımı için, SSF.net ise ağ bağlantıları ve düğüm ve bağlantı konfigürasyonlarını gerçekleştirmek için kullanılmaktadır (URL10). Simülatör için çalışmalar 1998 yılında başlamış 1999 yılında ilk sürümleri çıkarılmış ve 2004 yılında programın 2.0 versiyonu yayınlanmıştır. Programın bünyesinde saldırı ve güvenlik analizlerinin yapılması için ayrı modüller de bulunmaktadır. Literatürde SSFNet ilgili yapılmış akademik bir çok çalışma bulunmaktadır (Robert ve ark., 2003) (Nicol ve ark., 2003) (Yun ve ark., 2007) (Nicol ve ark., 2005).

3.9. GTnetS

GTNetS (The Georgia tech network simulator) büyük ölçekli bilgisayar ağlarında, değişik kriterler altında ağ üzerinde farklı simülasyon ortamlarının kurulması ve test edilmesi için geliştirilmiş bir simülatördür. GTnetS Georgia-Tech üniversitesinde Dr. George Riley tarafından açık kaynak kodlu bir simülatör olarak geliştirilmiştir. GTnetS ağ simülatörü Windows ve Unix paltformunda çalışabilir. GTnetS ağ simülatöründe amaç olabildiğince gerçek ortama yakın bir simülasyon ortamı oluşturmaktır (Riley, 2003). Ağ simülatöründe katmanlar arası protokol geçişleri net ve birbirinden tamamen ayrık yapıda tasarlanmıştır. Ağ modelleri C++ dilinde kodlanmıştır ve nesneye dayalı yaklaşım kullanılmaktadır. Bu yaklaşım simülatör üzerinde tasarım ve uygulama işlemlerinin daha kolay bir şekilde yapılmasını sağlar. GTnetS ağ simülatörü çok fazla sayıda uygulama ve protokolü desteklemektedir. Ayrıca simülasyonlar paralel ve dağıtık ortamlarda da gerçekleştirilebilir. GTnetS yönlendirme algoritması olarak Nix-Vector algoritmasını kullanır (Riley ve ark., 2000) ve bu algoritmanın kullanımı simülasyon sırasında bellek kullanım miktarını düşürerek, etkin bir kullanım sağlar. Paralelleştirme ve kullanılan yönlendirme algoritması simülatöre yüksek ölçeklenebilirlik sağlar. GTnetS ağ simülatörü, sahip olduğu grafiksel arayüzü sayesinde gerçekleşen simülasyonun görüntülenmesini

sağlar. Grafiksel arayüz kullanıcılarına, simülasyon nesnelerini aktif ve pasif etme imkanı sunar ayrıca simülasyon ile ilgili performans ve istatistiksel sonuçların görüntülenme ve incelenmesi mümkün olur. GTnetS ağ simülatörü ile ilgili detaylı bilgi Dr. George Riley tarafından hazırlanan simülatör kullanımı ile ilgili dökümandan ve resmi web sitesinden elde edilebilir (URL11). Literatürde GTnetS ağ simülatörü ile ilgili yapılmış akademik çalışmalar bulunmaktadır (Cheng ve ark., 2006) (Riley ve ark., 2004) (Maeda ve ark., 2005) (Li Xiao ve ark., 2007).

3.10. Ağ Simülatörleri Genel Karşılaştırma

Tablo 3.2. Bazı ağ simülatörlerinin karşılaştırılması

ÖZELLİK	NS-2	NS-3	Pdns	OPNET	OMNeT ++	SSFNeT	GloMo Sim	DEVS-Suite
Amaç	Eğitim, arştırma	Eğitim, araştırma	Eğitim, araştırma	Ticari	Eğitim, araştırma	Ticari	Özel amaçlı	Özel amaçlı
Model kütüphanesi	Güçlü	Güçlü	Güçlü	Güçlü	Güçlü	Zayıf	Orta	Zayıf
Analiz	Orta	Orta	Orta	Çok güçlü	Zayıf	Zayıf	Güçlü	Çok güçlü
Esneklik	Orta	İyi	Orta	İyi	Çok iyi	İyi	İyi	Çok güçlü
Dokümantasyon	Orta	Güçlü	Orta	Çok güçlü	Güçlü	Zayıf	Orta	Orta
Kullanım kolaylığı	Zor	Zor	Zor	Kolay	Orta	Zor	Zor	Kolay
Kullanıcı ara yüzü	Zayıf	Zayıf	Zayıf	Güçlü	Güçlü	Güçlü	Güçlü	Güçlü
Ölçeklenebilirlik	Orta	Çok iyi	Çok iyi	Orta	İyi	Çok iyi	İyi	İyi
Performans	Güçlü	Güçlü	Çok güçlü	Orta	Orta	Çok güçlü	Orta	Çok güçlü
Ağ Modeli	WAN	Büyük ölçek	Büyük ölçek	LAN, Uydu	LAN, MAN, Wireless	Büyük ölçek	LAN, MAN, WAN	LAN, MAN, WAN
Lisans	Açık kaynak	Açık kaynak	Açık kaynak	Ticari	Açık kaynak	Açık kaynak	Açık kaynak	Açık kaynak
Programlama dili	C++ ve TCL	C++ ve Python	C++ ve TCL	C++	C++	Java, C++	C++	Java
Platform	Unix, Linux	Unix, Linux	Unix	Xwind	Win.	Linux, Win.	Unix, Win.	Linux, Win.

Tablo 3.2'de yaygın olarak kullanılmakta olan ağ simülatörleri, kullanım amacı, model kütüphanesi, analiz, esneklik, dökümantasyon, kullanım kolaylığı, kullanıcı ara yüzü, ölçeklenebilirlik, ağ modeli, performans ve platform bakımından kıyaslanmaktadır. Bu tablo farklı çalışmalarda elde edilen sonuçlar değerlendirilerek

ortaya çıkarılmıştır (Rahman ve ark., 2009) (Zeigler ve ark., 2002) (Fujimoto ve ark., 2003) (Waupotitsch ve ark., 2006). Yapılan değerlendirmeler sonucunda, NS-3 ağ simülatörünün, performans, ölçeklenebilirlik, dokümantasyon ve model kütüphanesi, Pdns ağ simülatörünün; performans ve ölçeklenebilirlik, OPNET ağ simülatörünün; analiz, dokümantasyon, kullanım kolaylığı, kullanıcı ara yüzü kriterlerine göre daha ön plana çıktığı söylenebilir. SSFNET ağ simülatörünün büyük ölçekli ağların simülasyonunda daha verimli olabileceği ve ölçeklenebilir bir yapıya sahip olduğu, DEVS-Suite ağ simülatörünün ise kullanıcı ara yüzü, analiz, esneklik ve performans noktalarında iyi sonuçlar ürettiği tespit edilmiştir.

BÖLÜM 4. NS-2 AĞ SİMÜLATÖRÜ

NS (Network Simulator), ağ simülasyonu oluşturmak ve gerçekleştirmek için ilk olarak 1989 yılında geliştirilmeye başlanmış, akademik araştırmalar için büyük öneme sahip açık kodlu bir ayrık olay ağ simülatörü olarak tanımlanmaktadır. NS kullanımı, 1995 yılında DARPA'nın sponsorluğunda bir ivme kazanmıştır ve günümüzde de simülatörün geliştirilmesi gönüllüler tarafından sürdürülmektedir.

NS ile kablolu ya da kablosuz ağlarda istenilen miktarda düğümler ve bu düğümler arası linkler tanımlanabilir, yönlendirme algoritmaları ile çoğa gönderim protokolleri kullanılabilir ve ad-hoc, WiFi, WiMax, vb. gibi bir takım popüler kablosuz ağ uygulamalarının modellemeleri ve simülasyonu gerçekleştirilebilir. NS simülatörü, ağ araştırma ve eğitimini destekleyerek protokol tasarımı, trafik araştırması sağlamakta, ücretsiz açık kaynak kodu ile karşılaştırmalı ve model paylaşımlı güvenilir bir deney ortamı sunmaktadır. NS-2 ağ simülatör programının kurulumu EK-1'de detaylı olarak verilmiştir. Mohammad, yapmış olduğu yüksek lisans tezinde, NS-2 ağ simülatörü kullanarak tasarsız ağlar için DSDV, AODV ve DSR yönlendirme protokollerinin performans karşılaştımasını yapmışlardır. Yüksek mobilite sık sık bağlantı hatalarına yol açar. Yeni yönlendirme bilgisi ile birlikte tüm düğümlerin güncellenmesi kapsamında yer alan protokol yükü, DSDV'de AODV ve DSR'ye göre çok daha fazladır. Çünkü AODV ve DSR'de rotalar sadece gerektiğinde oluşturulur. Çalışmada NS-2 simülatöründe oluşturulan ağ simülasyonu üzerinde, paket gecikme zamanları, ağ çıkışı ve rotalama üzerinde değerlendirmeler gerçekleştirilmiştir (Mohammad, 2006).

Dönertaş 2008 yılında yapmış olduğu yüksek lisans tezinde, tasarsız ağlarda uçtan uca tıkanıklık ve yerel çekişme problemleri, gerçek hayatta uygulanan örnek senaryolar ve çeşitli simülasyonlar kullanılarak araştırılmış, ayrıntılı performans değerlendirilmesi ve analizi yapılmıştır. Farklı ağ parametrelerinin değişen senaryo

koşullarındaki etkilerinin analiz edilmesi için simülasyonlar oluşturulmuştur. Bu çalışmada dört farklı ağ modelinde de rasgele yol modeli kullanılarak karşılaştırma yapılmıştır. Çalışmada simülasyonların gerçekleştirilmesi için NS-2 ağ simülatörü kullanılmıştır (Dönertaş, 2008).

Ezreik ve arkadaşları 2012 yılında yapmış oldukları çalışmada, kablosuz bilgisayar ağlarını sınıflandırmış ve kablosuz ağlarda tasarım ve simülasyon için NS-2 ağ simülatörünü kullanmışlardır. Oluşturulan simülasyonda testler gerçekleştirilmiş ve Network Animator programı kullanılarak simülasyon çıktıları elde edilmiştir. Simülasyon sırasında elde edilen ağ çıkış değerleri, paket kayıp ve düşme oranları değerlendirilip, grafiksel ortama aktarılarak değerlendirme yapılmıştır (Ezreik ve ark., 2012).

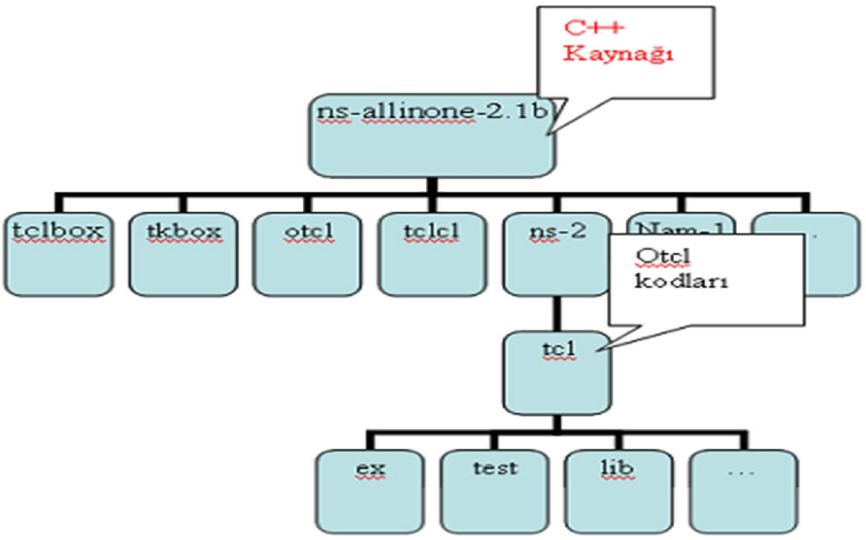
Micheal 2004 yılında gerçekleştirmiş olduğu tez çalışmasında, sensör ağlarında son zamanlarda kullanılan SENSE ağ simülatör programının genişletilebilirlik, ölçeklenebilirlik ve tekrar kullanılabilirlik açısından NS-2 ağ simülatörü ile karşılaştırılması gerçekleştirilmiştir. NS-2 ve SENSE ağ simülatörlerinin bileşen modelleri verilmiş ve programlar tanıtılmış ve bellek yönetim sistemleri ve öğrenebilirlik açısından incelenmiştir. Daha sonra belirlenen simülasyonlar üzerinde testler yapılarak, elde edilen sonuçlar üzerinde değerlendirme yapılmıştır (Micheal, 2004).

Feng 2004 yılında yapmış olduğu yüksek lisans tezinde NS-2 ağ simülatörü üzerinde BGP-4 protokolünün tasarım ve uygulamasını gerçekleştirmiştir. BGP protokolü internet gibi geniş ölçekli ağ yapıları üzerinde kullanılan bir yönlendirme protokolüdür. Bu sebepten dolayı BGP'nin performansı direk olarak iletimin kalitesini ve değişik parametreleri etkilemektedir. Simülasyonlar üzerinde teorik bilgilerden çok uygulama sonucu elde edilen dinamik ve pratik sonuçlar elde edilmektedir. Gerçekleştirilen uygulamada BGP-4 yapısı ile ns-BGP modeli geliştirilmiş ve rota seçimi, tekrar bağlantı kurulumu ve rota yansıtımı işlemleri gerçeklenmiş ve ns-BGP 'nin ölçeklenebilirlik analizi yapılmıştır (Feng, 2004).

4.1. NS-2 Mimari Yapısı

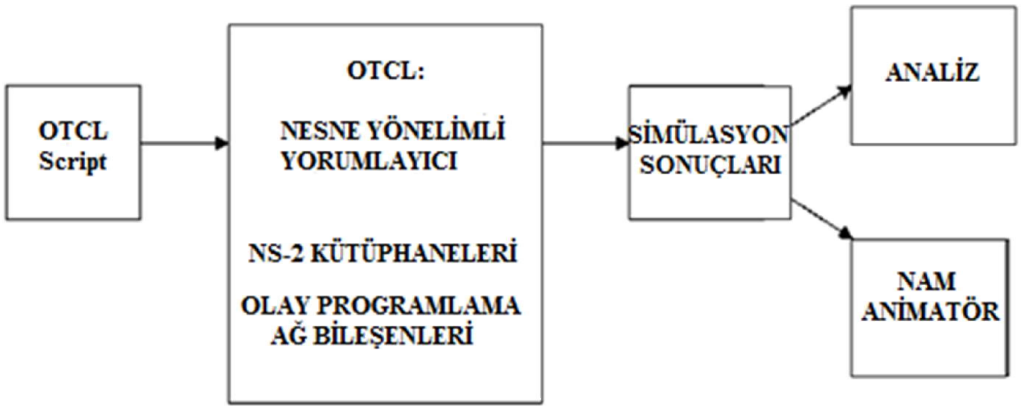
NS programlama, yapılan çalışmalarda belli bir algoritmaya uygun olarak oluşturulmaktadır. Öncelikle olay programlayıcısı kurulur, izleme açılır ve kapanır, ağ modeli oluşturulur, ilgili yönlendirme protokolü kurulur, sisteme hatalar girilir, iletim bağlantısı ile trafik oluşturulur ve uygulama gerçekleştirilerek veriler iletilir. NS simülatörünün ikinci versiyonu NS-2 olarak adlandırılmaktadır. NS, C++ tabanlı bir simülatördür ve Tcl dilinin objeye yönelik bir versiyonu olan OTcl ile bir simülasyon arayüzü vasıtasıyla ağ simülasyonu gerçekleştirmek mümkündür. C++, her paket işlemi için hızlı, detaylı bir kontrol sağlarken, Otcl ise mevcut C++ nesnelerini kullanarak simülasyon senaryo ayarları, periyodik veya tetiklemeli tahrik ve kolay yazılım imkanlarıyla öne çıkmaktadır. Şekil 4.1'de görüldüğü gibi, C++ ve Otcl aynı sınıf hiyerarşisini kullanmaktadır.

NS-2, Linux platformunda çalışmaktadır. Cygwin programı vasıtasıyla Microsoft Windows ortamında simülasyonlar gerçekleştirilebilmektedir. NS-2 simülatörünün son sürümü olan 2.35, Ekim 2011'de kullanılmaya başlanmıştır. NS-2'nin ağ modellemesindeki en önemli dezavantajı düz evren modeli kullanmasıdır. Ancak, gerçek ağlar engebeli yer yüzü şekilleri, yükselti, yansıma, gecikme ve gölgeleme gibi etkiler nedeniyle idealden uzak bir görünüme sahiptir. Bu duruma çözüm olarak oluşturulan gölgeleme etkisi modelinin ise çok etkin çalışmadığı görülmüştür. NS-2 yerel, geniş ve kişisel alan ağları ile ilgili simülasyon çalışmalarında kullanılmaktadır. Şekil 4.1'de NS-2 bileşenleri ve TCL ilişkisine ait şekil görülmektedir.



Şekil 4.1. NS-2 Bileşenleri ve TCL ile olan ilişkisi

Şekil 4.2'de NS-2 ağ simülastörü üzerinde işlemlerin gerçekleşmesini gösteren akış diagramı görülmektedir. Üretilen OTCL script, nesne yönelimli yorumlayıcı ve NS-2 kütüphanelerinin kullanılması ile simülasyon sonuçları üretilmektedir. Simülasyon sonuçları kullanılarak, analiz işlemleri ve Nam animatör programı kullanılıp gerçekleşen simülasyon grafik ortamında görsel olarak izlenebilir.



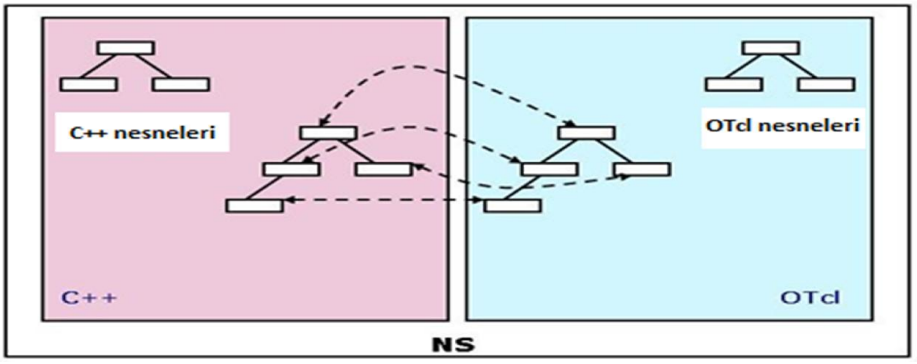
Şekil 4.2. NS-2 ağ simülatörü olay akış diagramı

NS-2 ağ simülatör programı bir ağ oluşturmayı ve gerçekleştirilen simülatör ortamında bağlantı analizlerini yapmayı sağlayan bir simülatordür. NS-2 dinamik ve elverişli bir network oluşturmaya yardımcı olur. NS-2 ayrık tabanlı bir ağ simülatörüdür. Basit ağ protokollerinin modellenmesine odaklanır. NS-2 ağ

simülatörü nesne yönelimli bir yazılım içerir ve modüler bir yapıya sahiptir. NS-2 ağ simülatörünün yardımcı programları NAM ve TCL programlarıdır. TCL, NS-2'nin derlendiği bir derleyici programıdır ve NS-2 de veya TCL de yazılan kodlar bu derleyici tarafından derlenir. NS-2 ağ simülatörü, internet protokollerinin tasarımının gerçekleştirilmesini sağlar. TCP protokolleri ile ilgili uygulamalarda yaygın olarak kullanılmaktadır. NS-2 internet katmanında farklı yeni protokoller oluşturmak için, internet protokollerinin etkileşimlerini göstermek, internetteki yeni mimari yapıların performanslarını belirlemek ve etkilerini göstermek için kullanılmaktadır. NS-2' de programlama yapılırken aşağıdaki adımlar izlenir:

1. Olay programlayıcı kurulumu,
2. İzlemeyi açma ve kapama,
3. Ağı oluşturma,
4. Yönlendirmeyi kurma,
5. Hatalar girme,
6. İletim bağlantısı oluşturma,
7. Trafiği oluşturma,
8. Uygulama-düzeyli veri iletimi.

Şekil 4.3'te NS ağ simülatörü üzerinde C++ ve OTCL dili arasındaki nesnesel ilişki görülmektedir.

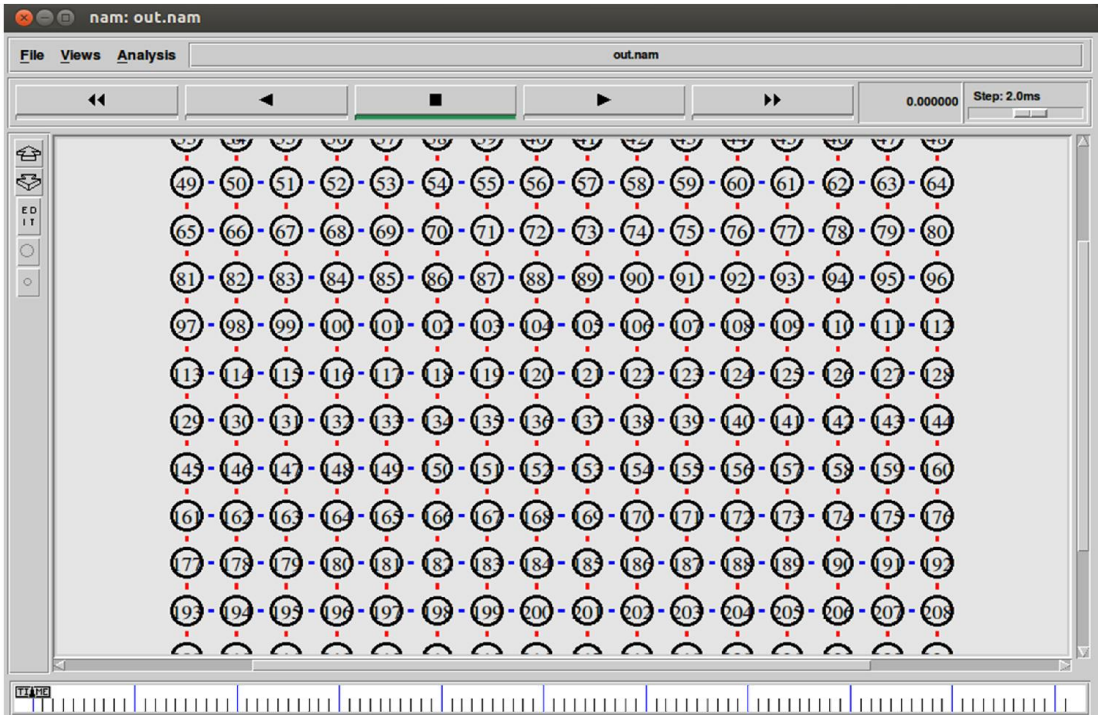


Şekil 4.3. NS simülatörünün C++ ve OTcl nesne ilişkisi

4.2. NS-2 Ağ Simülatöründe Simülasyon Sonuçlarını İzleme

Nam network simülasyon izlerini ve gerçek dünya paket izleri verilerini izlemek için bir animasyon tabanlı araç olan Tcl/TK'dır (URL13). Nam, topoloji düzeni, paket seviye animasyon ve çeşitli veri kesif araçlarını destekler. Nam LBL' de başlamıştır. Nam için geliştirme olanakları hala desteklenen bir proje olan VINT projesi ile sağlanmaktadır.

Nam'ı kullanmak için ilk adım bir trace dosyası oluşturmaktır. Trace dosyası topoloji bilgilerini içermelidir. Örnek olarak düğümler, linkler ve paketlerdir. NS-2 simülasyonu boyunca kullanıcı, topoloji konfigürasyonu, düzen bilgisi ve paket izlerini izleme olaylarını kullanarak üretebilir. Trace dosyası üretildiği zaman, simülasyon izlenmek için hazır demektir. Çalışmaya başlaması üzerine, nam trace dosyasını okur, topolojiyi oluşturur ve gerçekleşen simülasyon gösterilir. Nam kullanıcı ara yüzü ile animasyonun birden fazla görünümünü sağlar. Nam düğüm, bağlantı, kuyruk, paket, temsilci ve monitor yapı bloklarını kullanarak animasyon yapar. Şekil 4.4'de NAM da bir ağa ait simülasyon görüntüsü bulunmaktadır.

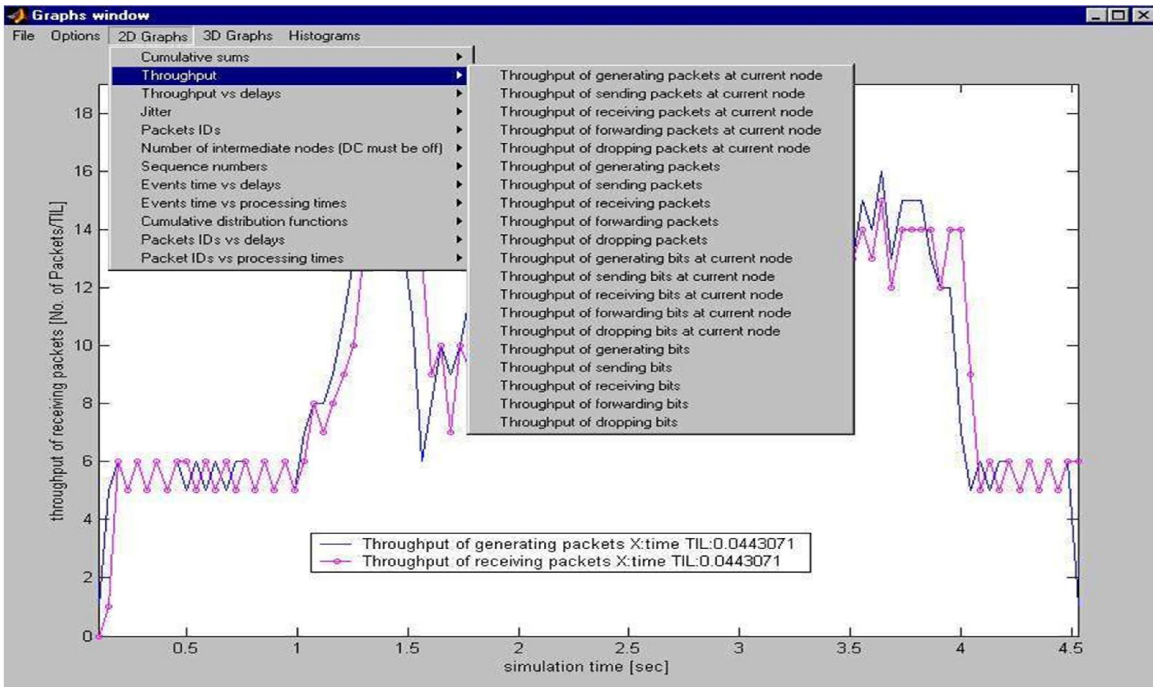


Şekil 4.4. NAM simülayon görüntüsü

4.3. NS-2 Ağ Simülatöründe Sonuçların Analizi

Tracegraph programı, NS-2 ağ simulator programı tarafından üretilen iz dosyalarını kullanarak, gerçekleştirilen simülasyona ait grafikler oluşturmak için kullanılan Matlab tabanlı bir grafik programıdır. Tracegraph programı tarafından oluşturulan grafikler kullanılarak, simülasyon sonuçlarının çok kolay bir şekilde analizi gerçekleştirilebilir. Tracegraph programı ile kablolu, kablosuz ağ ortamında bir çok protokole ait grafiksel sonuçlar üretilebilir. NS-2 programına gerekli eklentiler yapılarak, simülasyon sonucunda tr uzantılı dosyanın oluşturulması sağlanmaktadır. Simülasyon sonuçlarını içeren tr uzantılı dosya tracegraph programı kullanılarak açıldığında simülasyon ile ilgili bir çok istatiksel ve grafiksel veriler oluşturulabilmektedir (URL14). NS-2 programının tr uzantılı dosyayı üretmesi için, programa eklenmesi gerekli kod parçaçığı aşağıda verilmiştir. Şekil 4.5 'te tracegraph programına ait grafiksel ara yüz görülmektedir.

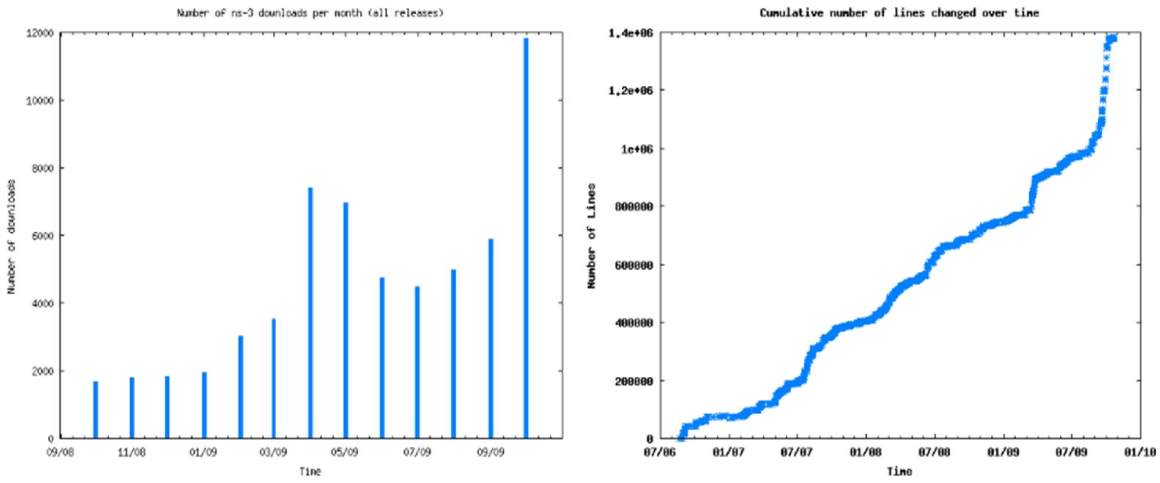
```
set izleme [open starizleme.tr w]
$ns trace-all $izleme
proc finish {} {global ns izleme
$ns flush-trace
close $izleme
exit 0 }
```



Şekil 4.5. Tracegraph programına ait ekran çıktısı

BÖLÜM 5. NS-3 AĞ SİMÜLATÖRÜ

NS-3 araştırmacılar tarafından hazırlanmış araştırma ve eğitimsel amaçlı bir ayrık olay ağ simülatörüdür. NS-3'ün yazılımı C++ ve Python programlama dillerinde yapılmıştır. 2006'da geliştirilmeye başlayan NS-3 projesi açık kodlu bir projedir. NS-3 simülatörü, NS-2'nin genişletilmiş hali değildir, yeni bir simülatördür. NS-3 kullanıcı tarafından değiştirilebilme ve genişletilebilme özelliğine sahiptir. Araştırmacıların, yeni model geliştirmelerine, var olan bir modelin hatalarını bulma ve geliştirmelerine ve sonuçlarını paylaşmalarına sürekli katkı sağlamaya dayanmaktadır. Şekil 5.1'de, yıllara göre NS-3 simülatörünün toplam indirilme sayısı ve 2006 yılından bugüne kadar kodda değiştirilen kümülatif satır sayısı gösterilmektedir. Bu grafikler incelendiğinde NS-3 ağ simülatörünü kullanma sayısında ciddi bir artış olduğu görülmektedir.



Şekil 5.1. NS-3 simülatörü indirilme sayısı ve kodda değiştirilen satır sayısı

NS-3 Unix ve Linux tabanlı sistemler üzerinde ve aynı zamanda Windows sistemi üzerinde çalışan bir programdır. Kullanıcılar için planlanmış python script arayüzü ile C++ dilinde yazılmıştır. Asıl olarak IPv4 ve IPv6 tabanlı ağlar üzerine

odaklanmıştır. Aynı zamanda sensörler ya da IP tabanlı olmayan IP mimarilerini de desteklenir. NS-3 aşağıdaki özellikleri desteklemektedir.

1. Sanal ağların yapısı ve olay zamanlayıcı, topoloji üreteçleri, zamanlayıcılar, rastgele değişkenler gibi nesneler için destek ve internet tabanlı ve diğer paket ağ sistemleri odaklı destekli diğer nesneler için ayrık olay ağ simülasyonu desteği sağlama.
2. Ağ simülasyon desteği, simülator işlemleri için gerçek ağ paketlerini alma ve yayma.
3. Dağıtık simülasyon desteği, çoklu işlemciler ya da dağıtık sistem olan yapılarda simülasyon gerçekleştirebilme.
4. Ağ simülasyonları için animasyon desteği.
5. Simülasyon çıkışında kayıt tutma, izleme istatistikleri için destek sağlar.

Henderson ve arkadaşları 2006 yılında gerçekleştirilen çalışmada, NS-3 ağ simülatörünün geliştirilmesine ait tasarlanan projede yapılacak olan çalışmaları ve NS-3 ağ simülatörünün yapısı, mimarisi ve özellikleri hakkında bilgiler verilmiştir. NS-3 projesi ile ortaya çıkarılması düşünülen ürün ve beklentiler açıklanmıştır. NS-2 ağ simülatörünün genel yapısı aktarılmış, devamı olmadığı ifade edilmiş ve NS-3 ağ simülatörünü farklı kılacak olan özellikleri belirtilmiştir. Katmansal bazda NS-2 ağ simülatöründe varolan protokol ve mimari yapıları, NS-3 ağ simülatöründe eklenecek olan yeni mimari ve protokol yapıları açıklanmıştır. NS-3 ön plana çıkaracak olan özellikler olan ölçeklenebilirlik, verimlilik vb. kriterlerden bahsedilerek, gelişim süreci planları aktarılmıştır (Henderson ve ark., 2006).

Wang ve arkadaşları 2009 yılında, ağ eğitiminde NS-3 ağ simülatörünün kullanımı ile ilgili bir çalışma gerçekleştirmiştir. Ağ yapılarının karmaşılığından dolayı, ağ eğitimlerinde teorik metodlardan çok, pratik ve uygulamaya yönelik olan eğitimsel araçların kullanımı uygundur. Bu çalışmada NS-2 ve NS-3 ağ simülatörleri hakkında kısa bilgi verilmiş, NS-3'ün tercih edilmesini sağlayan özellikleri belirtilmiştir. Eğitimde NS-3 kullanılması için bir uygulama geliştirilmiş, ve uygulamada kullanılacak olan kodlar verilerek açıklama yapılmıştır. Uygulamada paket akışının

kontrolü ve paket içeriklerinin incelenmesi için WireShark programı kullanılmıştır (Wang ve ark., 2009).

Jonathan ve arkadaşları 2010 yılında yapmış oldukları çalışmada, dinamik bilgisayar ağ yapılarının modellenmesi için kullanılan karınca kolonisi optimizasyon sisteminin uygulama ve simülasyonunu gerçekleştirmiştir. Uygulamada ağ simülatörü olarak NS-3 kullanılmıştır. Gerçekleştirilen tasarımda NS-3'te karınca kolonisi sistemi için paket formatı oluşturulmuştur. Yönlendirme algoritması ve farklı parametreler kullanılarak sistem üzerinde simülasyon ve uygulamalar yapılmıştır. Ayrıca sistemde ağ simülatörü olarak NS-3'ün yanında kıyaslama için NS-2 kullanılarak elde edilen sonuçlar karşılaştırılmıştır (Jonathan ve ark., 2010).

Nicola ve arkadaşları 2011 yılında yayınlanan makalelerinde, LTE ağlarının simülasyonlarında kullanılmak üzere NS-3 ağ simülastöründe bir modül tasarımı gerçekleştirmişlerdir. Yapılan çalışmada, dizayn kriterleri, MAC fonksiyonlarını uygulanması gerçekleştirilerek, test ve değerlendirme aşamasında farklı algoritmaların kullanımı ile elde edilen test sonuçları sunulmuştur. Ayırca simülasyonlar, çalışma zamanı ve bellek kullanım miktarI değerlerine göre de incelenmiştir (Nicola ve ark., 2011).

Mathieu 2010 yılında doktora tezinde yaptığı çalışmada, ağ yapıları üzerinde çalışan bir çok kimse tarafından araştırma konusu olmuş optimizasyon ve akış kontrolü üzerinde bir çalışma gerçekleştirmiştir. Özellikle büyük çaplı ağlarda bu konu çok daha büyük öneme sahiptir. Büyük ölçekli ağların simülasyonlarının gerçekleştirilmesi ve optimizasyon işlemlerinin tasarım aşamasında gerçekleşmesi için NS-3 ağ simülatörü üzerinde yeni bir araç tasarımı gerçekleştirilmiştir. Öncelikle, NS-3 çekirdeğine gerçek zamanlı simülasyon kabiliyetini sağlayacak ve gerçek dünya ile bağlantı kuracak bir yapı entegre edilmiştir. Oluşturulan araç birçok karışık mimarinin gerçek zamanlı simülasyonunu gerçekleştirerek sonuçların incelenmesine olanak sağlamaktadır. NS-3 ağ simülatörüne entegre edilen bir başka araç ise direk olarak kodu çalıştırabilecek bir modüldür. İki modül sayesinde büyük ölçekli ağlar üzerinde anlık olarak işlemler gerçekleştirilebilmekte ve optimizasyon

işlemleri yapılabilmektedir. Çalışmada NEPI adı verilen bir çerçeve tasarımı gerçekleştirilmiştir (Mathieu, 2010).

5.1. NS-3 Mimarisi

NS3 projesinde Waf sistem mimarisi kullanılmıştır. Waf, Python temelli yeni jenerasyon bir sistemdir. NS-3, NS-2'nin şu anki sahip olduğu modellerin hepsine sahip değildir, fakat NS-3 IP adreslemeyi, daha çok internet protokolleri tasarımları ve daha detaylı 802.11 modeli vb. kullanarak düğümler üzerinde çoklu ara yüzleri hatasız işlemek gibi özellikleri destekler.

NS-3 simülatörü, NS simülatörünün 3. versiyonu olarak 1 Temmuz 2006 tarihinden itibaren geliştirilmeye başlanmıştır ve projenin tamamlanması için 4 yıllık bir süre öngörülmüştür. Günümüzde tasarımı devam etmekte olan NS-3 simülatörü, Washington üniversitesi, Georgia Teknoloji Enstütüsü, ICSI ve INRIA gibi kurumlar tarafından geliştirilmeye devam etmektir. NS-3 simülatörü araştırma, geliştirme ve akademik faliyetlerde kullanılmak üzere özellikle internet tabanlı sistemler için geliştirilen bir ayrık olay ağ simülatörü olarak tanımlanmaktadır. GNU GPLv2 lisanslı yani bedelsiz ve açık kaynak kodlu bir yapıya sahiptir.

NS-3, C++ ve Python dilleri kullanılarak yazılmaktadır. Yine NS-2 simülatöründe olduğu gibi Linux işletim sistemi üzerinde çalıştırılmaktadır ve Cygwin programı vasıtasıyla Windows tabanlı sistemlerde de kullanılabilmektedir. NS-3'te kod yapıları Doxygen isimli bir yazılım dökümantasyon program vasıtasıyla uygulanmaktadır. NS-3 simülatörünün birçok farklı sürümü mevcuttur. İlk istikrarlı sürüm olan NS-3.1, Temmuz 2008'de kullanıma sunulmuştur. Bu sürümde simülatör çekirdeği, TCP/IPv4 trafiği, noktadan noktaya CSMA ve WiFi modelleri mevcuttur. İkinci sürüm NS-3.2, Eylül 2008'te kullanıma sunulmuş ve Python bağlayıcıları, gerçek zaman programlama, IEEE 802.1D, istatistik ve nsn iskeleti oluşturulmuştur.

NS-3.3, Aralık 2008'de hazırlanmış ve emülasyon özelliği, ilk IPv6 ve ICMP standartları simülatöre kazandırılmıştır. Nisan 2009'da devreye alınan NS-3.4 sürümünde ise simülatöre obje isimlendirme, yeni WiFi modelleri ve kuyruk

listeleme fonksiyonları kazandırılmıştır. Temmuz 2009'da çıkartılan NS-3.5 sürümünde, IEEE 802.11e MAC EDCA, 802.11n A-MSDU, 802.11b PHY, Nakagami kayıp modeli, Gamma, Erlang ve Zipf rasgele değişken yapıları sisteme ilave edilmiştir. Ekim 2009'daki son NS-3.6 sürümünde ise, akış izleme, 5/10 MHz kanal ve WiFi fonksiyon eklemeleri, ICMP, IPv6 radvd, yeni test çerçevesi ve 802.11s mesh yapı ilaveleri gerçekleştirilmiştir.

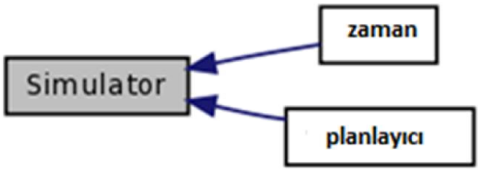
NS 3.7 ve NS 3.12 sürümleri arasında IPv4 broadcast datagram 1 den 64'e kadar değiştirilmiştir. Topoloji başlıkları eklenmiş, IPv6 desteği geliştirilmiş. AODV uzaklık vektöründe RFC 3561 geliştirilmiştir. Vimax modelinde Point-to-Multipoint (PMP) mod ve the WirelessMAN-OFDM PHY katmanıyla birlikte 802.16 özelliğinin MAC ve PHY gerçekleşimi sağlanmıştır. Kuyruk davranışları değiştirilmiştir. OFDM modülasyon tipi için Wi-Fi hata oranı değiştirilmiştir. Wi-Fi iletim oranlarının adı ve yapısı değiştirilmiştir. C++'ın desteklediği 64 bitlik sayı yapısını da desteklemeye başlamıştır. NS-3.13 de hareketlilik modülü iki nesneyle birlikte eklenmiştir. Bu modül Model::GetRelativeSpeed() metodudur. Yeni Ipv6AddressGenerator sınıfı ön ek ve arabirim kimliği tabanıyla birlikte ardışıl adres üretmek için eklenmiştir.

NS-3.17 Mayıs 2013 tarihinde çıkarılmıştır. LTE ve TCP ye ait farklı özelliklerle birlikte, daha bir çok modül üzerinde güncellemeler gerçekleştirilmiştir. En son sürüm olarak NS-3.19 Aralık 2013 tarihinde kullanıma sunulmuştur. NS-3 kütüphanesi Şekil 5.2'de gösterildiği gibi bir takım modüllerin birleşiminden oluşmaktadır. Bu modüller, çekirdek, simülatör, ortak-yaygın kullanım alanı, düğüm ve araçlar olarak sıralanabilir.

<table>
<tr><td colspan="3">Yardımcı</td></tr>
<tr><td>Yönlendirme</td><td>İnternet yığını</td><td>Sürücüler</td></tr>
<tr><td colspan="2">Düğüm</td><td>Mobil</td></tr>
<tr><td colspan="2">Simülatör</td><td>Ortak</td></tr>
<tr><td colspan="3">Çekirdek</td></tr>
</table>

Şekil 5.2. NS-3 simülatör modülleri

SIMULATOR (Simülatör): src/simulator dizininde bulunur ve olay zamanlama özelliklerini içerir. Aşağıda simülator işbirliği şeması gösterilmiştir.

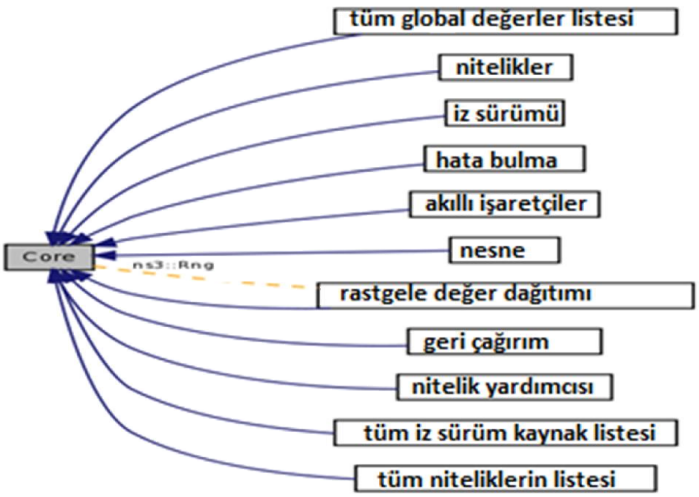


Şekil 5.3. NS-3 Simülatör modülü

Simulatör modülü ns3::Time, ns3::Scheduler ve ns3::Simulator sınıflarının içerir.

1. ns3::Time : Zamanı tutmak ve çeşitli zaman birimlerini birbirine çevirmek için kullanılan bir zaman yönetim sınıfıdır.
2. ns3::Scheduler : Olay zamanlayıcılar ile yeni simülasyon gerçekleştirmek için kullanılan temel bir olay zamanlayıcı sınıfıdır.
3. ns3::Simulator : Zamanlama ve iptal olaylarını oluşturmak için kullanılan simulator sınıfıdır.

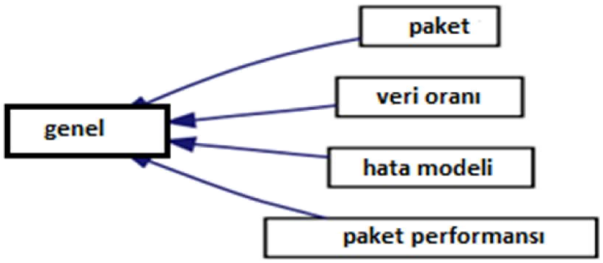
CORE (Çekirdek) : src/core dizininde bulunur ve diğer hiçbir modüle ihtiyaç duymayan bir dizi özellik içerir. Bu özelliklerin bazıları işletim sistemi bağımlıdır. Aşağıda çekirdek işbirliği şeması gösterilmiştir.



Şekil 5.4. NS-3 Çekirdek modülü

Çekirdek modülü; bir test yöneticisi ve test sınıfı (ns3::Test and ns3::TestManager), hata ayıklama özellikleri (Logging, Assert), bir functor sınıfı (ns3::Callback), rastgele değişken dağıtıcısı, saat işlevine erişmek için işletim sisteminden bağımsız bir arayüz (ns3::SystemWallClockMs), izleme kaynaklarını desteklemek için bir temel sınıf nesnesi (ns3::ObjectBase), referans sayma ve dinamik nesne toplamayı desteklemek için bir sınıf nesnesi (ns3::Object), ns3::Object sınıfı ile birlikte çalışacak şekilde tasarlanmış bir akıllı pointer sınıfı (ns3::Ptr), bir simülasyondaki bütün özellikleri ayarlamak, kontrol etmek ve kaynakları izlemek için kullanılan bir yapılandırma sınıfı (ns3::Config) barındırır.

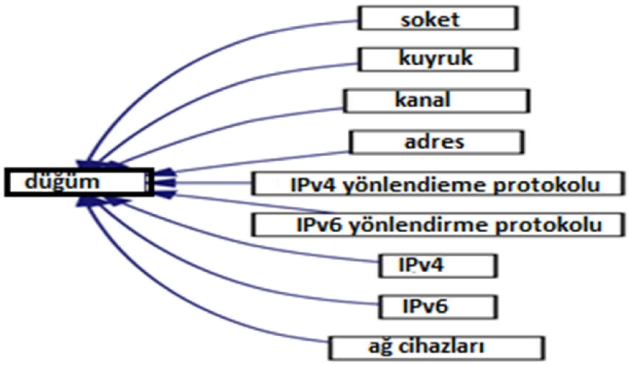
COMMON (Genel): src/common dizininde bulunur ve birçok ağ bileşenine ayrılmış şekilde ağ simülasyonlarına has özellikler barındırır. Aşağıda genel işbirliği şeması gösterilmiştir.



Şekil 5.5. NS-3 Genel modülü

Genel modülü simülasyon paketleri oluşturmak ve manipüle etmek için bir paket sınıfı (ns3::Packet, ns3::Header, and ns3::Trailer) barındırır.

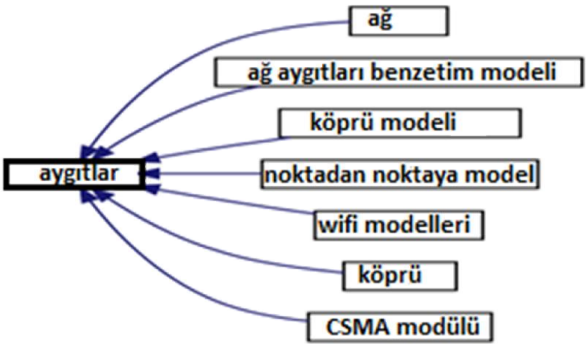
NODE (Düğüm): src/node dizininde yer alır ve her düğüm (daha özelde IPv4 düğümleri) tarafından gerçekleştirilmesi gereken özet arabirimleri tanımlar. Aşağıda düğüm işbirliği şeması gösterilmiştir.



Şekil 5.6. NS-3 Düğüm modülü

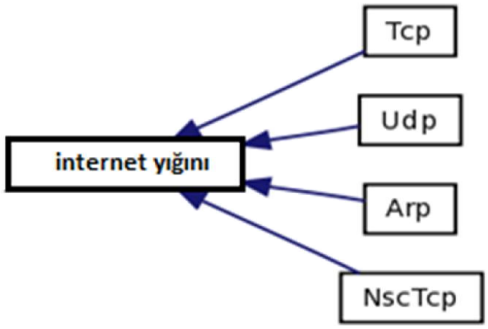
Node (Düğüm) modülü; herhangi bir yeni tip ağ düğümünün alt sınıflandırması için bir temel sınıf (ns3::Node), IP layer protokolünden MAC layer protokolüne kadar özetler içeren modeller (ns3::NetDevice, ns3::Channel) ve uygulama katmanı API özetlerini içeren modelleri (ns3::Application, ns3::Socket, ns3::SocketFactory ve ns3::Udp) barındırır.

DEVICES (Sürücüler): src/devices dizininde yer alır. Bir dizi MAC düzeyli modelleri barındırır. Aşağıda sürücüler işbirliği şeması gösterilmiştir.



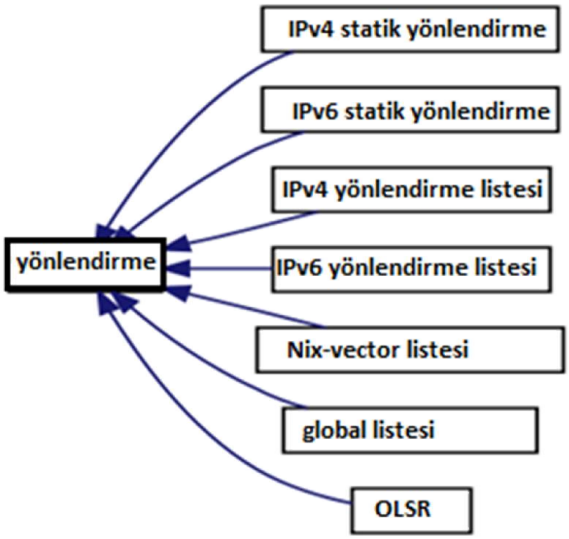
Şekil 5.7. NS-3 Aygıtlar modülü

INTERNETSTACK (İnternet-Yığın): Internet-Yığın modülü bir IPv4 stack, ARP modülü, UDP\TCP gerçekleştiricisi barındırır. Aşağıda internetstack işbirliği şeması gösterilmiştir.



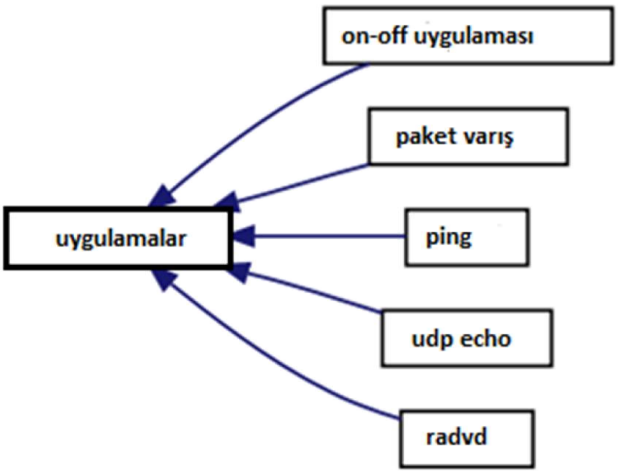
Şekil 5.8. NS-3 İnternet-Yığın modülü

ROUTING (Yönlendirme): Yönlendirme modülü yönlendirme işlemlerini gerçekleştirmede kullanılan alt modül ve sınıfları barındırır. Aşağıda yönlendirme işbirliği şeması gösterilmiştir.



Şekil 5.9. NS-3 Yönlendirme modülü

APPLICATIONS (Uygulamalar): Yönlendirme modülü yönlendirme işlemlerini gerçekleştirmede kullanılan alt modül ve sınıfları barındırır. ns3::Application sınıfı NS-3 uygulamaları için temel bir sınıf olarak kullanılabilir. Aşağıda uygulamalar işbirliği şeması gösterilmiştir.



Şekil 5.10. NS-3 Uygulamalar modülü

5.2. NS-3 Ağ Simülatörünün Kurulumu Ve Örnek Script Dosyasının Çalıştırılması

NS-3 ağ simülatör programının kurulumu EK-2'de detaylı olarak verilmiştir. Waf kontrolü sayesinde script dosyalarının çalıştırılması gerçekleştirilmektedir. Böylece paylaşılan kütüphane ve çalışma zamanında kullanılacak kütüphane yolları sistem tarafından yüklenmiş olur. Yazmış olduğunuz script dosyanızı çalıştırmak için terminalinize aşağıdaki komut satırının girilmesi gerekmektedir.

./waf --run hello-simulator

Waf öncelikle yazdığınız kodun doğru bir şekilde derlenenip derlenemeyeceğini kontrol eder ve ardından programı çalıştırır. Yukarıdaki kod satırını yazdığınızda karşınıza aşağıdaki gibi bir çıktı görüntülenecektir.

Hello Simulator

Bu çıktı elde edildiğinde, ilk script dosyası başarıyla çalıştırılmıştır. Eğer scriptinizi çalıştırdıktan sonra Hello Simulator çıktısı görüntülenmediyse, Waf yükleme modu "optimized" olarak ayarlanmış olabilir. Bu projedeki uygulanacak olan tüm script dosyaları çalıştırıldığında karşınıza çıktı görüntülerinin gelebilmesi için Waf yükleme modunu "debug" olarak ayarlamalıdır. Bu ayarlam için aşağıdaki kod satırının terminale yazılması gerekmektedir.

./waf -d debug --enable-examples --enable-tests configure

Bu kod satırını yazdıktan sonra Waf'a debug versiyonlarının çalıştırılması için *./waf* komutu yazılmalıdır.

5.3. NS-3 Script Kod Satırlarının İncelenmesi

NS-3 ağ simülatörünün kurulum ve konfigürasyonun ardından, ana dizinde bulunan repos dizini içerisinde ns-3-allinone dizini bulunmaktadır. Burada bulunan ns-3-dev examples/tutorial dizinine girdiğinizde first.cc dosyası bulunmaktadır. Bu script dosyası noktadan noktaya bağlı iki düğümü ve bu iki düğüm arasında echo paketinin gönderilmesini içermektedir. Adım adım her kod satırı detaylandırılarak anlatılacaktır.

~/repos/ns-allinone/ns-3-dev/examples/tutorial/first.cc dosyasını açtığınızda ilk satır emacs modunu belirtir. NS-3 simülatör kodları GNU Genel Kamu Lisansı ile lisanslanmıştır. Genelde kod sayfanızın ilk başında bu lisansla ilgili metni görebilirsiniz.

```
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
```

Her C++ kodlamasında olduğu gibi bu script dosyasının başındada program için kullanılacak kütüphanelerin çağrılması gerekmektedir. Bu işlem aşağıdaki gibi yapılmaktadır.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
```

Bir sonraki kod satırı ise NS-3 namespace tanımlamasıdır.

using namespace ns3;

NS-3 projesi C++ ile yapılandırılmış ve namespace ismi olarak NS-3 kullanılmıştır. Bir sonraki adımda ise loglamanın yapılması işlemi gerçekleştirilmektedir.
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

Kod satırlarına devam edildiğinde her C++ kodlamasındaki gibi bir program ana fonksiyonunun tanımlandığını görülmektedir.

```
int
main (int argc, char *argv[])
{
```

Bu kod satırları server ve istemci arasında loglamanın yapılabilmesi için gerekli tanımlamaları içermektedir.

```
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

Bu kod satırları sayesinde simülasyon süresinde gönderilen ve alınan paketler çıktı ekranında görüntülenecektir. Bu adımlardan sonra topolojinin oluşturulması ve simülasyonun çalıştırılması işlemlerine geçilecektir. Aşağıdaki iki satır ile iki adet NS-3 düğüm nesnesi oluşturulacaktır. Bu düğümler simülasyonda kullanılacak bilgisayarların simülasyonu için kullanılacaktır.

```
NodeContainer nodes;
nodes.Create (2);
```

Bu script dosyasında iki düğümden oluşan sunucu ve istemci arasında bir point-to-point (noktadan noktaya) iletişim söz konusu olacaktır. Bunu gerçekleştirmek için aşağıdaki kod satırları yazılmalıdır.

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

Buradaki ilk satır noktadan noktaya iletişimin sağlanması için gerekli yığını tanımlar. İkinci satırda ise oluşturulan nesnenin veri hızının 5Megabit per/second olarak ayarlanmasını göstermektedir. Üçüncü satırda ise gecikme değeri 2ms olarak ayarlanmaktadır. Bu adımlardan sonra önceden tanımlanmış olan iki düğümün bir container içerisine yerleştşirilmesi gerekmektedir. Bu işlem için aşağıdaki kod satırları yazılmaktadır.

```
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

Burada devices isimli Network Device Container nesnesi tanımlanıyor ve ardından noktadan noktaya düğümlerin yüklenmesi işlemi gerçekleştiriliyor. Böylece iki düğüm bir container içerisinde simülasyona hazır hale gelmiş olmaktadır. Henüz simülasyon için geçerli bir protokol tanımlamadık. Bu iki düğüm arasında Internet Protokol yığınını (TCP, UDP, IP vb.) kullanmak için aşağıdaki kod satırlarını eklemeniz gerekmektedir.

```
InternetStackHelper stack;
stack.Install (nodes);
```

İnternet protokolünü kullandığımızda her iki düğümün bir biri ile haberleşmesi için bir IP adresi tanımlanması gerekmektedir. Ip adres tanımlamalarının yapılması için aşağıdaki kod satırlarının yazılmış olmalıdır.

Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");

IP versiyon 4 adresleme kullanılarak 10.1.1.0 dan başlayacak şekilde ve alt ağ maskesi 255.255.255.0 olacak şekilde IP numaralarının tahsisi yapılacaktır. Bir sonraki kod satırında IP adreslerinin simülasyonumuzda hazırladığımız düğümlere tahsis edilmesi işlemi gerçekleştirilecektir.

Ipv4InterfaceContainer interfaces = address.Assign (devices);

Bu script dosyasında son olarak iki düğüm arasında bir uygulamanın tanımlanması ve simülasyonun çalıştırılması işlemleri gerçekleştirilecektir. Bu uygulamada UdpEchoServerApplication ve UdpEchoClientApplication kullanılacaktır. Uygulama da bir düğümün sunucu olarak atanması ve UdpEchosServerApplication bu düğüme atanması işlemi aşağıdaki gibi gerçekleştirilmektedir.

UdpEchoServerHelper echoServer (9);

Aşağıdaki kod satırlarını incelediğinizde bir düğümün sunucu olarak atanması ve sunucu üzerinde kurulan uygulamanın başlangıç ve bitiş süreleri belirtilmektedir.

ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

Sunucu tanımlamaları tamamlandıktan sonra diğer düğüme de istemci tanımlaması ve gerekli konfigürasyonların yapılması gerekmektedir. Bunun için aşağıdaki kod satırlarının yazılması gereklidir.

UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

```
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

İstemcinin ayarlamalarını başlangıç ve bitiş sürelerini tanımladıktan sonra simülasyonun çalıştırılması için aşağıdaki kod satırı yazılmaktadır.

```
Simulator::Run ();
Simulator::Destroy ();
  return 0;
}
```

Simülasyonun başlatılması ve daha sonra RAM bellekten kaldırılması için simülasyonu kaldırma (destroy) işlemi gerçekleştirilir. Ana fonksiyonumuzun tamamlandığını belirterek fonksiyona sıfır değeri geri döndürerek programımızı sonlandırıyoruz. Script dosyasının çalıştırılması için, öncelikle hazırladığınız script dosyanızı /examples/tutorial/first.cc dosyasını kopyalayarak ns-3-dev dizini altındaki scratch klasörüne kopyalayınız ve ismini myfirst.cc olarak değiştiriniz. Bu işlemi aşağıdaki kod satırlarıyla terminalinizi kullanarak da yapabilirsiniz.

```
cd ../..
cp examples/tutorial/first.cc scratch/myfirst.cc
```

Ardından scriptinizin derlenmesi için Waf komutu ile çalıştırılmalıdır.

```
./waf
```

Eğer kod doğru bir şekilde şekilde çalışmış ise, aşağıdaki şekilde bir ekran çıktısı üretmesi gerekmektedir.

```
Waf: Entering directory `/home/sau/repos/ns-3-allinone/ns-3-dev/build'
[614/708] cxx: scratch/myfirst.cc -> build/debug/scratch/myfirst_3.o
[706/708] cxx_link: build/debug/scratch/myfirst_3.o -> build/debug/scratch/myfirst
Waf: Leaving directory `/home/sau/repos/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (2.357s)
```

Script dosyanızın derlenmesi başarıyla tamamlandıktan sonra bu dosyayı çalıştırabilirsiniz. Bunun için aşağıdaki gibi Waf –run <dosyaismi> komutunu çalıştırabilirsiniz.

./waf --run scratch/myfirst

Script dosyanız çalıştırıldıktan sonra aşağıdaki gibi bir ekran çıktısı üretilmektedir.

Waf: Entering directory `/home/sau/repos/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/home/sau/repos/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (0.418s)
Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2

Burada ilk olarak istemci (10.1.1.1) den sunucuya (10.1.1.2) 1024 baytlık veri gönderimi gerçekleştiriliyor. Ardından sunucu 1024 baytlık veriyi aldığını belirten bir echo mesajı gönderiyor. Daha sonra sunucu istemciye 1024 baytlık veri gönderiyor. İstemci ise 1024 baytlık veriyi başarıyla aldığını belirtmektedir. Böylece simülasyon tamamlanmaktadır.

5.4. NS-3 Ağ Simülatöründe Sonuçları İzleme (NetAnim)

NetAnim (Network Animator) NS-3 ağ simülatöründe gerçekleştirilen simülasyonu çevrimdışı olarak izleyebildiğimiz bir ara yüz programıdır. NetAnim qt4 grafiksel kullanıcı arayüzü geliştirme programı tabanlı bir yazılımdır. NetAnim ilk versiyonu George F. Riley tarafından geliştirilmiş 2. ve 3. versiyonları John Abraham tarafından yazılmıştır (URL12). NS-3 ağ simülatörü gerçekleştirilen simülasyonu izlemek için XML uzantılı bir dosya üretmektedir. Üretilen bu dosya NetAnim programı ile çalıştırılabilir.

NS-3 ağ simülatöründe programın XML dosyası üretmesi için koda aşağıdaki eklentilerin yapılması gerekmektedir.

```
#include "ns3/netanim-module.h"
std::string animFile = "grid-anim.xml";
cmd.AddValue ("animFile", "File Name for Animation Output",
animFile);
AnimationInterface anim (animFile);
```

NetAnim programını kurulumu şu şekilde gerçekleştirilmektedir:
http://www.nsnam.org/wiki/index.php/NetAnim#Downloading_NetAnim
adresinden NetAnim indirilir. Daha sonra uçbirim(terminal) açılarak sırasıyla aşağıdaki işlemler gerçekleştirilir.

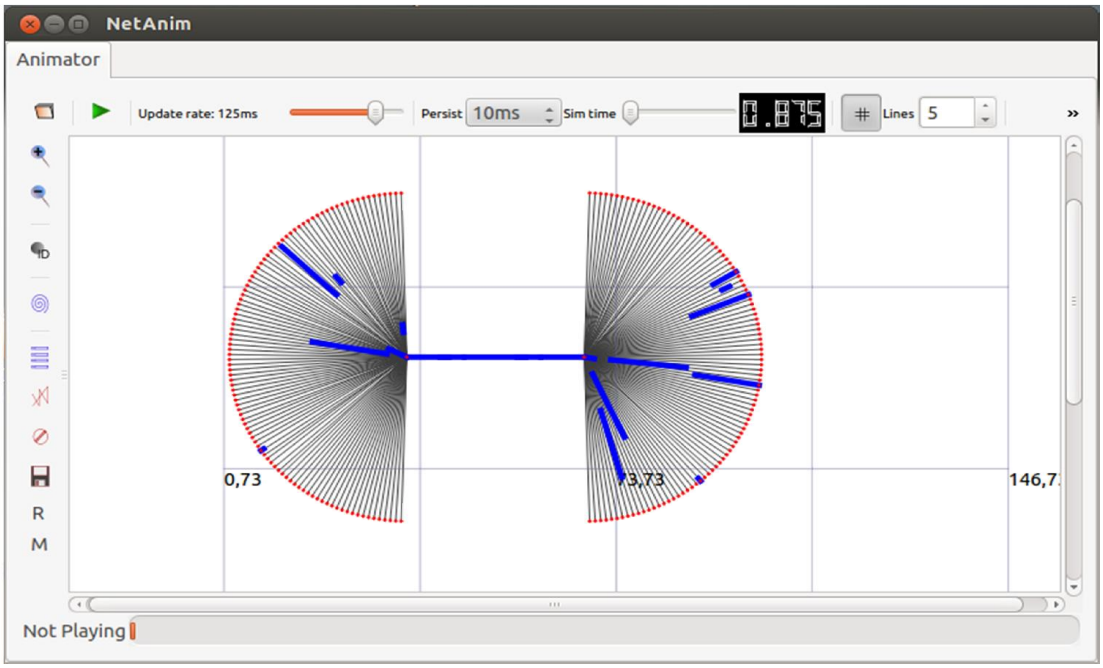
1. tar -xzvf NetAnim. tar. gz

2. cd NetAnim

3. sudo apt-get install qt4-dev-tools

4. qmake qt4

5. make

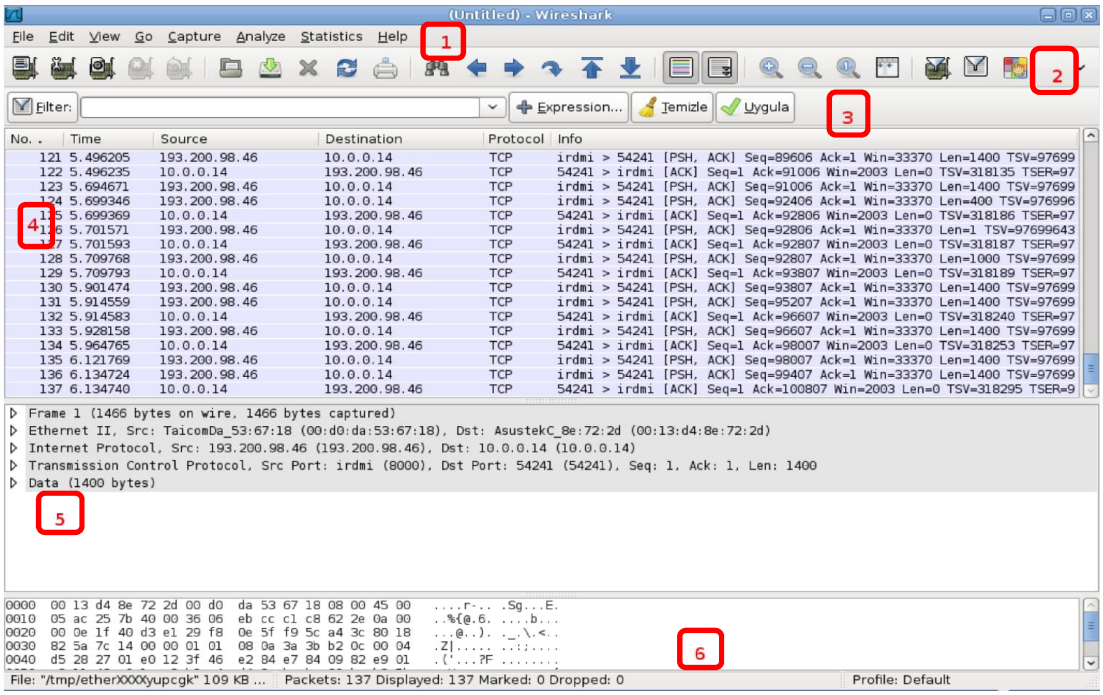


Şekil 5.11. NetAnim'de simülasyon görüntüsü

Şekil 5.11' de NS-3 ağ simülatöründe oluşturulan xml dosyasının çalıştırılması sonucu gerçekleşen simülasyona ait görsel arayüz görülmektedir.

5.5. NS-3 Ağ Simülatöründe Sonuçların Analizi

Wireshark çok güçlü analiz özelliklerine sahip, açık kaynak kodlu bir paket analiz yazılımıdır (URL15). Wireshark, bilgisayar ağınıza herhangi bir saldırı olması durumunda uyarma özelliği yoktur. Fakat sıradışı bir durum meydana geldiğinde analiz işlemlerini gerçekleştirip, problemin ne olduğunu farketmemizi sağlar. NS-3 ağ simülatörü, simülasyon sonucunda oluşan sonuçların analizini gerçekleştirmek için pcap uzantılı dosyalar üretmektedir.



Şekil 5.12 Wireshark ekran görüntüsü

Şekil 5.12 ' de wireshark programına ait ekran görüntüsü bulunmaktadır. Program üzerinde yer alan kısımlar şöyledir.

1. Menü çubuğu 2. Araç çubuğu 3. Görüntüleme filtresi çubuğu 4. Özet alanı

5. Protokol ağacı alanı 6. Veri alanı

Wireshark programında pcap dosyasının üretilmesi için aşağıdaki kod parçacığının yazılan NS-3 simülasyon koduna eklenmesi gerekmektedir.

```
AsciiTraceHelper ascii;
Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream
("grid.tr");
p2p.EnableAsciiAll (stream);
csma.EnableAsciiAll (stream);
p2p.EnablePcapAll ("gridd");
csma.EnablePcapAll ("grid", false);
```

BÖLÜM 6. NS-2 ve NS-3 PERFORMANS TESTLERİ VE KARŞILAŞTIRMA

Bu bölümde NS-2 ve NS-3 ağ simülatörleri kullanılarak, yıldız ve ızgara topolojileri üzerinde performans testleri gerçekleştirilmiştir. Tablo 6.1' de simülasyon sırasında kullanılan bilgisayarlara ait donanımsal konfigürasyonlar verilmiştir. Simülasyon testleri her iki bilgisayar üzerinde ayrı ayrı çalıştırılarak, simülasyon sonuçları elde edilmiş ve değerlendirme yapılmıştır.

Tablo 6.1. Simülasyonda kullanılan bilgisayar konfigürasyonları

Bilgisayar 1	**Bilgisayar 2**
2 GB RAM Intel® Core™ 2 Duo İşlemci 1,86 GHz işlemci hızı	4 GB RAM Intel® Core™ i5 İşlemci 2,30 GHz işlemci hızı

6.1. Izgara Topoloji Simülasyonu

Bu simülasyonda 16x16'lık ızgara modeli hem NS-2 hem de NS-3 simülatörleri kullanılarak, iki farklı bilgisayarda çalıştırılmış ve performans ölçümleri yapılmıştır.

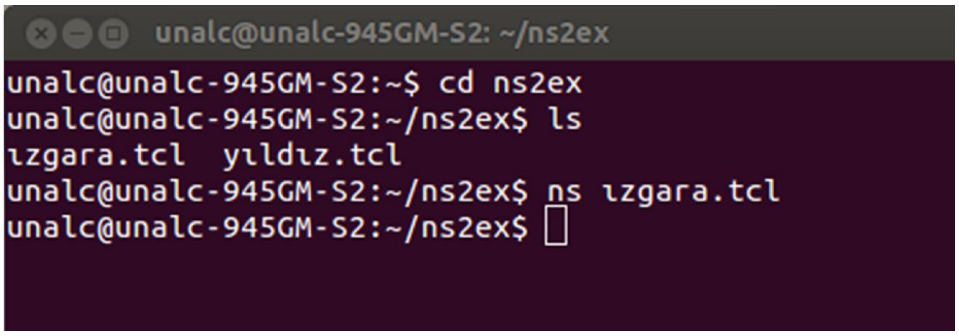
Tablo 6.2. Izgara simülasyonu parametreleri

PARAMETRE	**DEĞER**
Protokol	UDP
Bant Genişliği	5 mbps
Gecikme	2 ms.
Simülasyon Süresi	10 sn
Düğüm Sayısı	256
Paket boyutu	512 mb.
Veri hızı	500 kb/s
Kuyruk yapısı	Drop-tail

Izgara ağ modelinde kullanılan parametreler; protokol, bant genişliği, gecikme zamanı, simülasyon süresi, düğüm sayısı, paket boyutu, veri hızı ve kuyruk yapısına ait veriler Tablo 6.2'de verilmiştir.

6.1.1. NS-2 Ağ simülatöründe ızgara topoloji simülasyonu

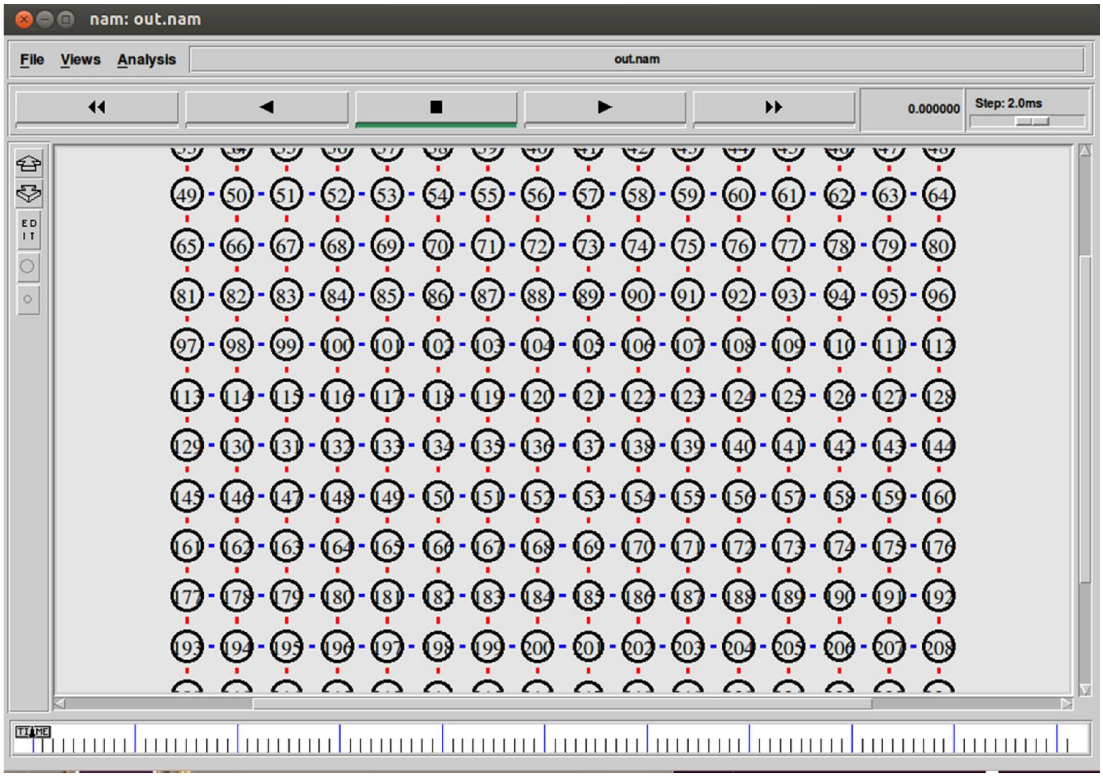
Izgara topolojinin NS-2 ağ simülatörü üzerinde gerçeklenmesi için kullanılan kod EK-3'te verilmiştir. NS-2 ağ simülatöründe, simülasyonun çalıştırılması için konsolda Tcl uzantılı dosyanın bulunduğu konuma gelinerek, ns komutu ile simülasyon gerçekleştirilebilir. Şekil 6.1'de gerçekleştirilen işleme ait ekran çıktısı görülmektedir.



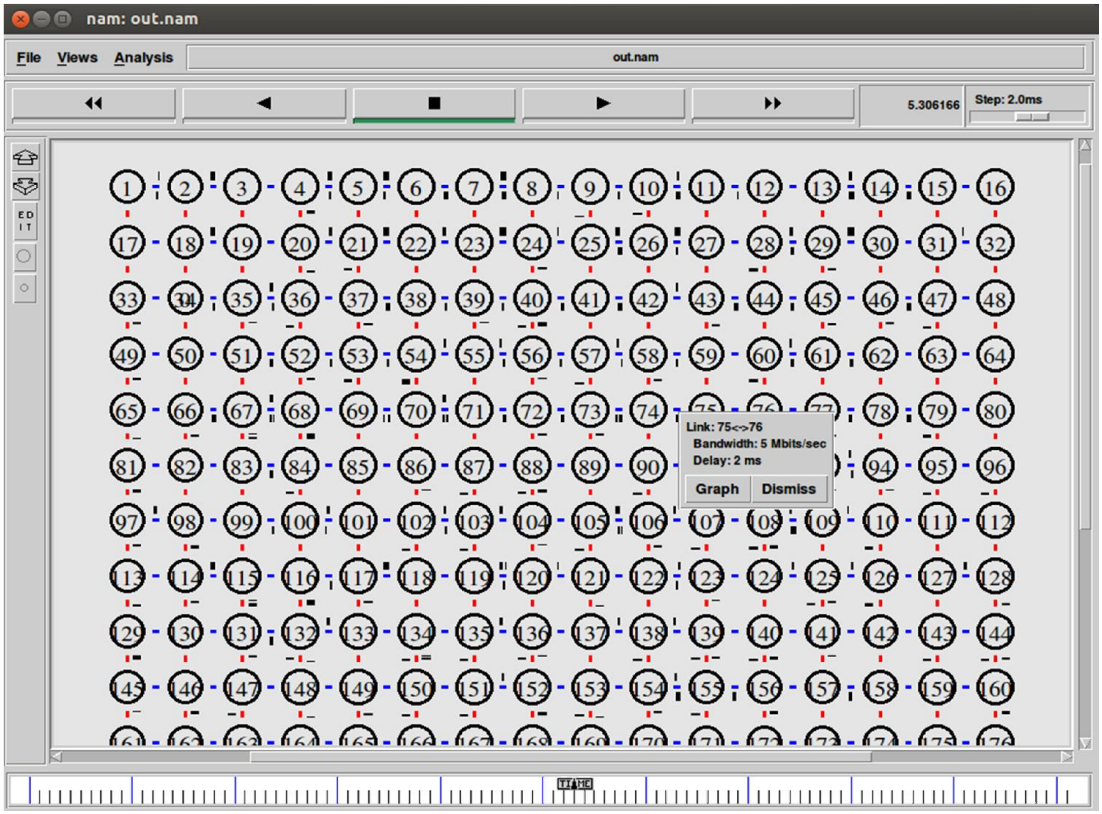
Şekil 6.1. Terminalde NS-2 simülatörü çalıştırma komutu

6.1.1.1. NS-2 ızgara topolojisi simülasyon çıktıları

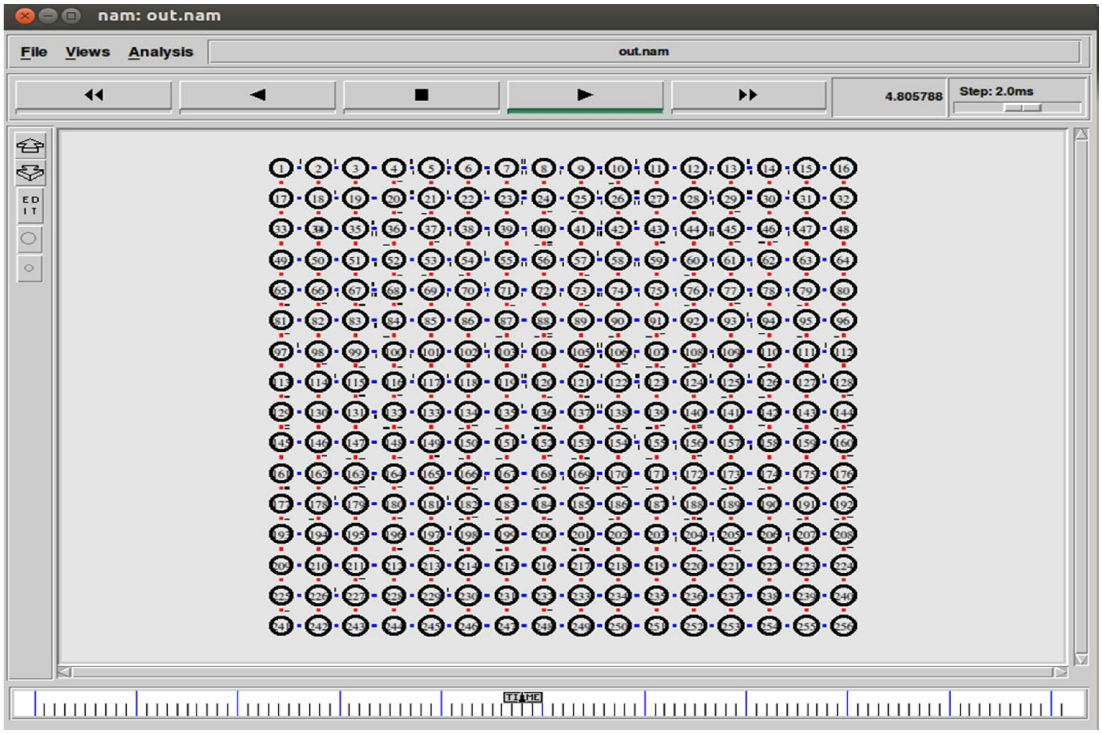
Şekil 6.2, 6.3 ve 6.4'te NS-2 ağ simülatörü üzerinde gerçekleştirilen ızgara ağ simülasyonuna ait, Network Animatör (NetAnim) programı kullanılarak elde edilen simülasyon çıktıları görülmektedir. Şekil 6.2'de simülasyon başlamadan önceki görüntü ve Şekil 6.3 ve 6.4'te simülasyonun gerçekleşmesi sırasında kaydedilen görüntü bulunmaktadır.



Şekil 6.2. Bilgisayar 1'de NS-2 ızgara görüntüsü



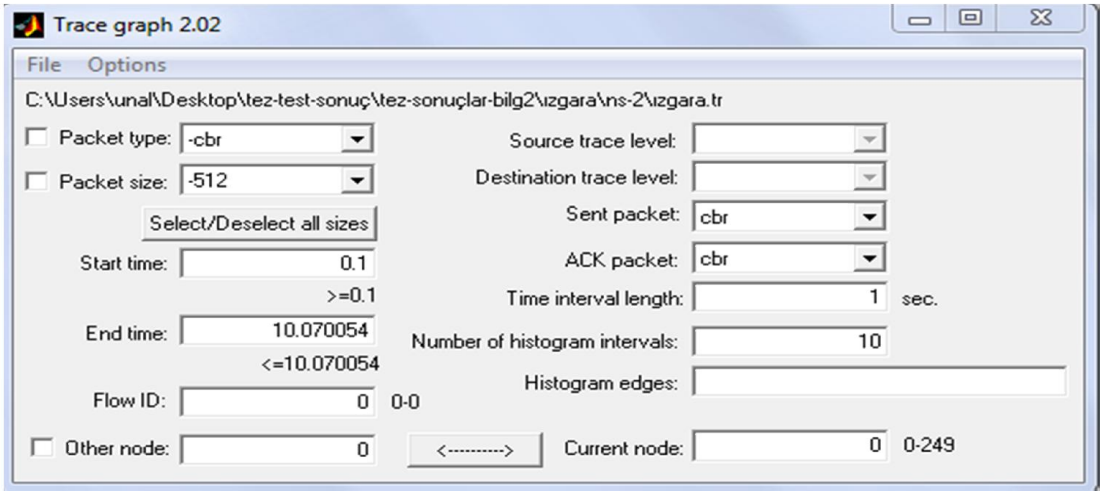
Şekil 6.3. Bilgisayar 1'de NS-2 ızgara simülasyonunun 5.3 sn' deki trafik durumu



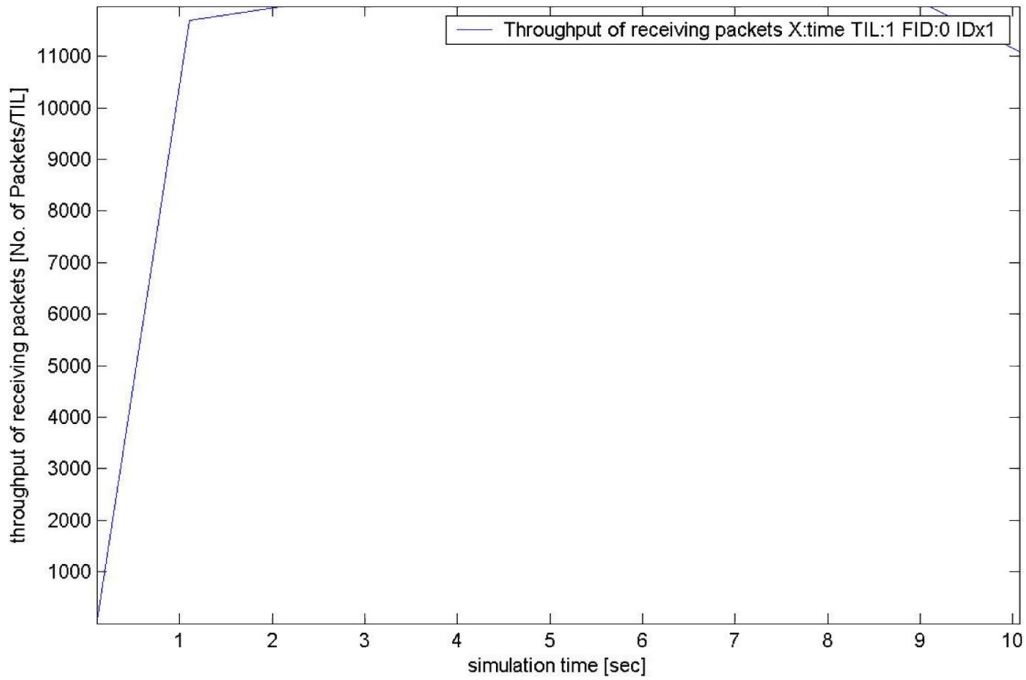
Şekil 6.4. Bilgisayar 2'de NS-2 ızgara simülasyonunun 4.8 sn' deki trafik durumu

6.1.1.2. NS-2 ızgara topoloji ağ çıkışı değerleri

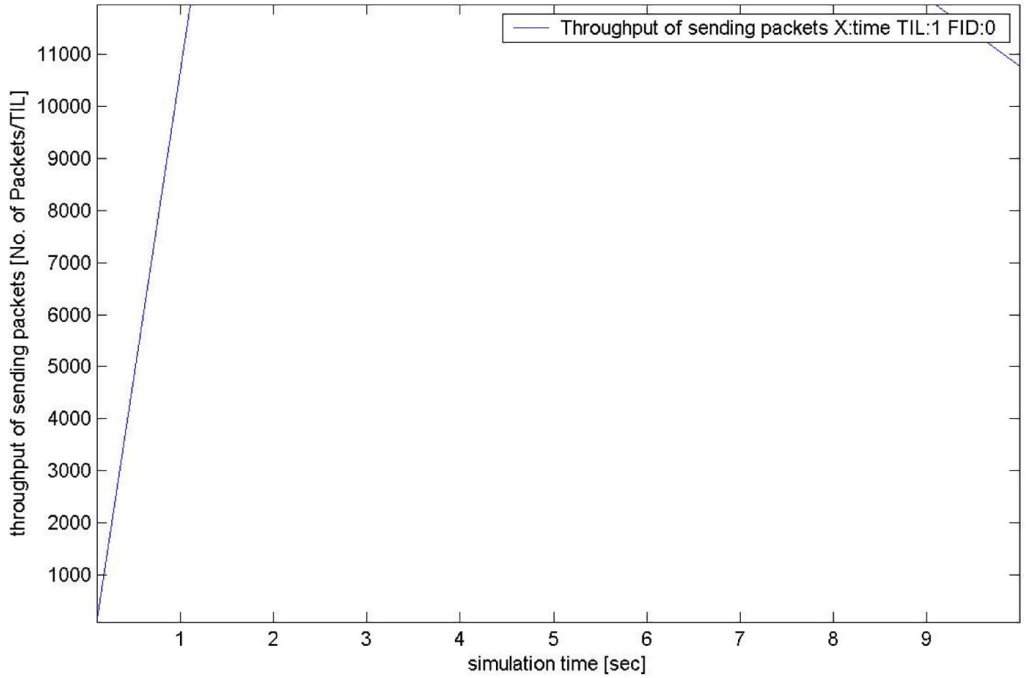
Izgara ağ simülasyonunun NS-2 ağ simülatörü üzerinde çalıştırılması sonucu elde edilen tr uzantılı iz dosyasının Tracegraph programına yüklenmesiyle ağ simülasyonu ile ilgili birçok istatiksel sonuçlar elde edilebilmektedir. Şekil 6.5'te Tracegraph programına ait ana ekran görüntüsü görülmektedir.



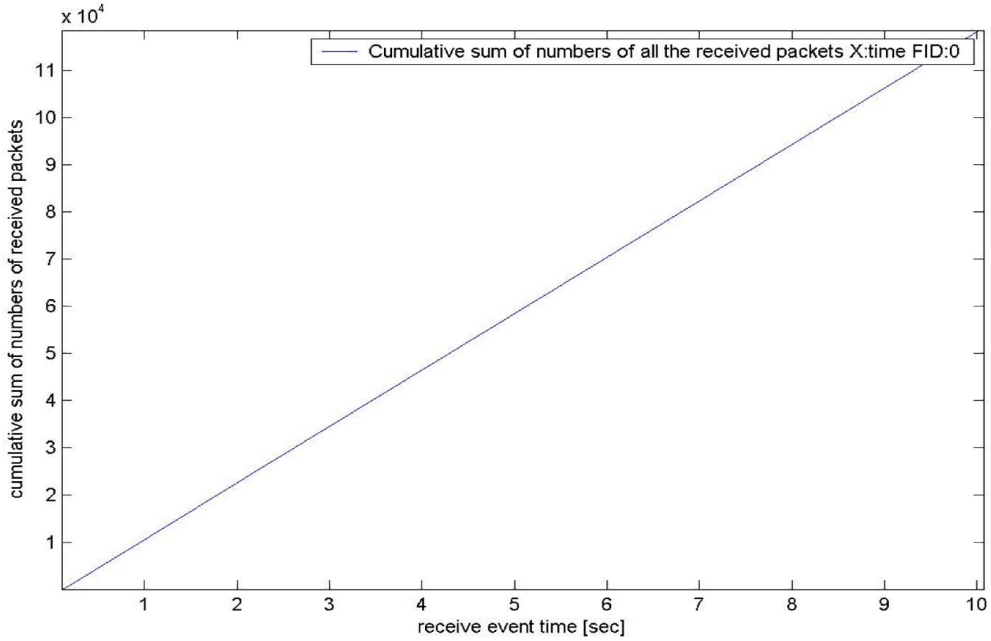
Şekil 6.5. Trace graph programına ait ana ekranı



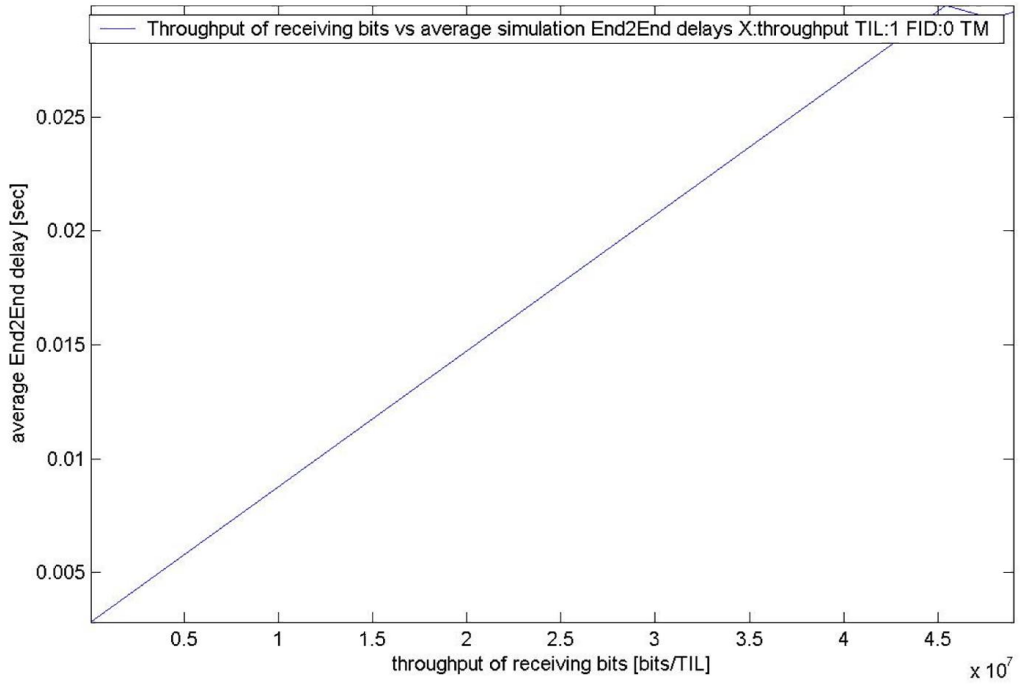
Şekil 6.6. Simülasyon zamanı ve alınan paketlerin ağ çıkış değerleri



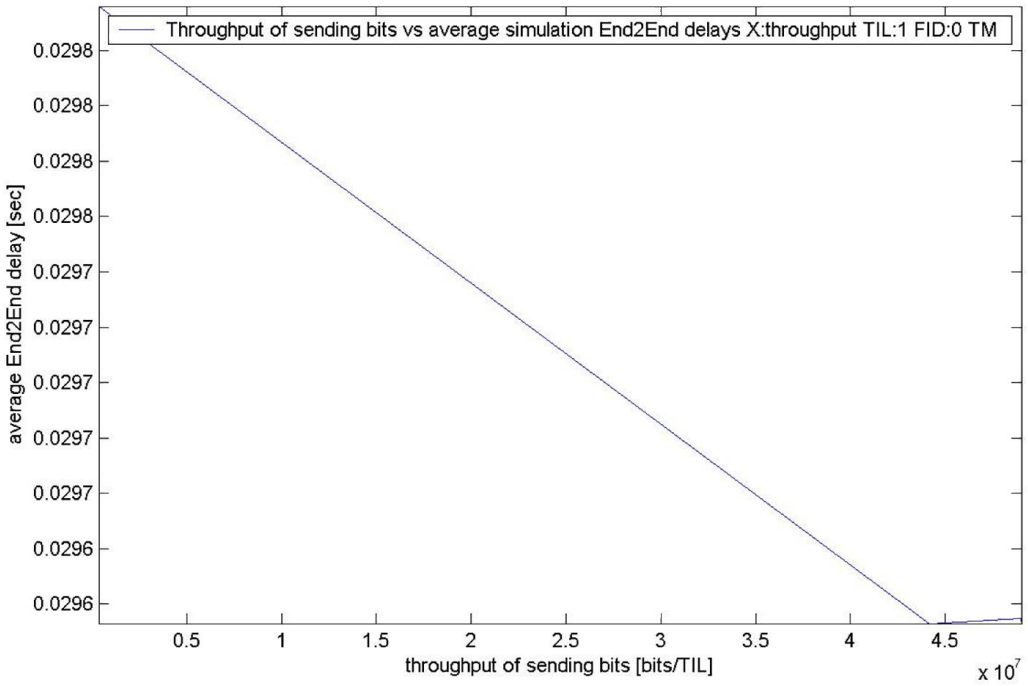
Şekil 6.7. Simülasyon zamanı ve gönderilen paketlerin ağ çıkış değerleri



Şekil 6.8. Simülasyon paket alım zamanı ve alınan paketlerin toplamı



Şekil 6.9. Simülasyon alınan bitlerin ağ çıkışı ve ortalama gecikme değerleri



Şekil 6.10. Simülasyonda gönderilen bitlerin ağ çıkışı ve ortalama gecikme değerleri

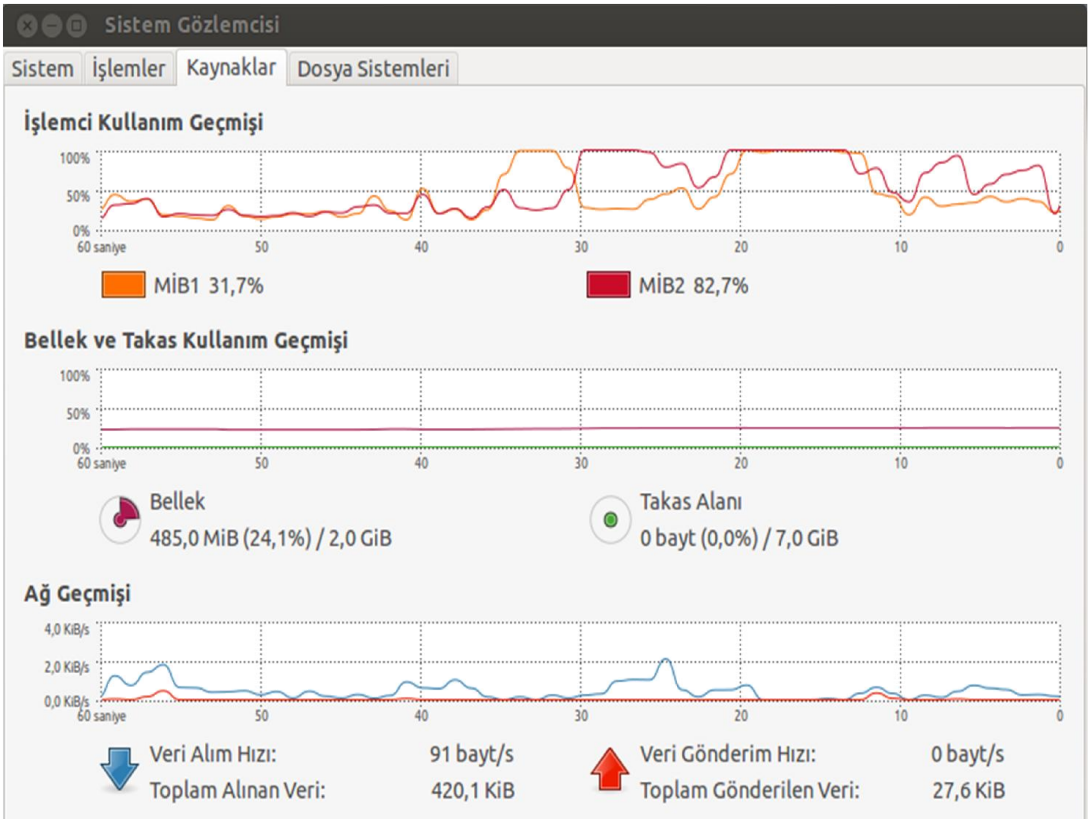
Şekil 6.6'da simülasyon zamanı boyunca alınan paketlerin ağ çıkış değerlerine ait grafik görülmektedir. Grafikte 1 ve 2. sn'de yükselen bir grafik seyreden değer 2-10 sn. arası max. değere ulaşmış ve bu şekilde devam etmiştir. Şekil 6.7'de simülasyon zamanı boyunca gönderilen paketlerin ağ çıkış değerlerine ait grafik görülmektedir. Bu grafik alınan paketlerin ağ çıkış değerleri ile yakın değerler taşımaktadır.

Şekil 6.8'de simülasyon zamanı boyunca paket alımının zamana bağlı değişimi ve alınan toplam paket sayılarına ait grafik görülmektedir. Simülasyon süresi boyunca alınan toplam paket sayısı doğrusal olarak artış göstermiştir.

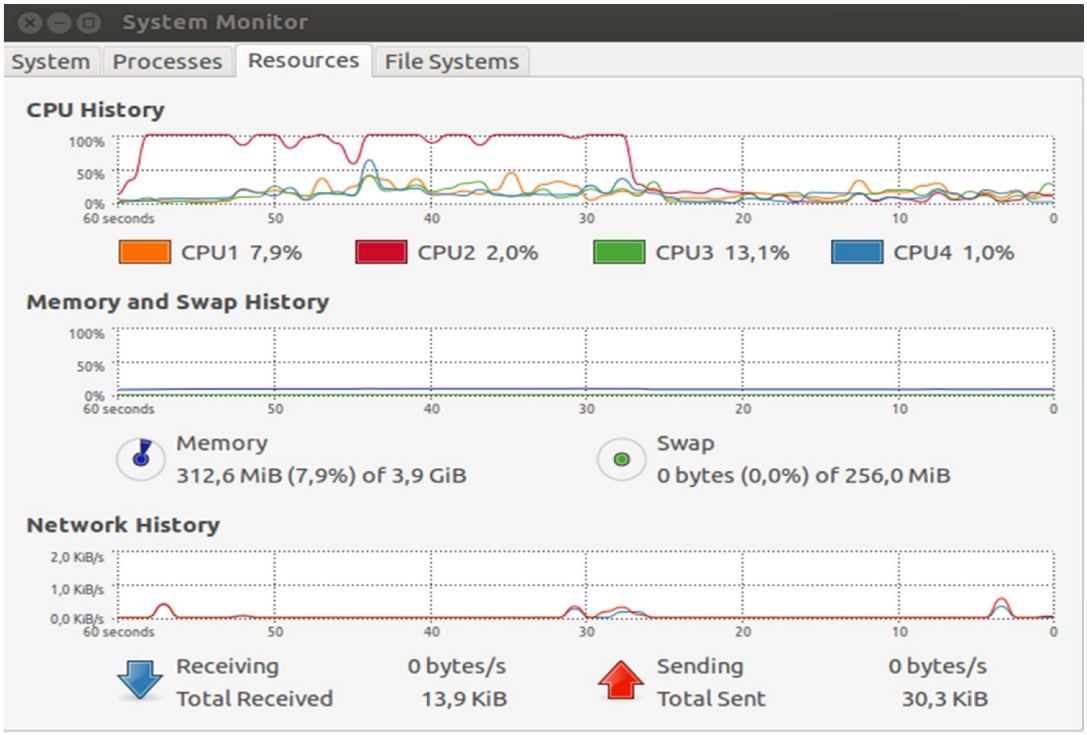
Şekil 6.9 ve 6.10'da simülasyon zamanı boyunca alınan ve gönderilen bitlere ait ağ çıkış değerlerinin noktadan noktaya ortalama gecikme zamanı değerleri görülmektedir. Alınan ve gönderilen bitlerde gecikme zamanı birbirine zıt değerlerde değişim göstermişlerdir. Alınan bitlerin ağ çıkış değerlerine ait ortalama gecikme zamanı doğrusal bir artış gösterirken, gönderilen bitlerin ağ çıkış değerlerine ait ortalama gecikme zamanı doğrusal bir azalış göstermiştir.

6.1.1.3. NS-2 ağ simülatöründe cpu ve belek kullanımı

Şekil 6.11 ve 6.12'de NS-2 simülatörü üzerinde ızgara simülasyonun çalıştırılması sırasında bilgisayar1 ve bilgisayar2'deki işlemci ve bellek kullanımı gösterilmiştir. Bu işlemlerin gerçekleştirilmesi sırasında Ubuntu sistem gözlemcisinden yararlanılmıştır. Bilgisayar1 ve bilgisayar2'de farklı işlemci sayıları mevcuttur. Grafikte simülasyon süresince işlemcilere ait kullanım yüzdeleri, bellek kullanım miktarları, ağ üzerindeki trafik miktarına ait bilgiler görülmektedir. Bilgisayar1 ve bilgisayar2'de işlem zamanı boyunca 1 adet işlemcinin %100 oranında kullanıldığı tespit edilmiştir.



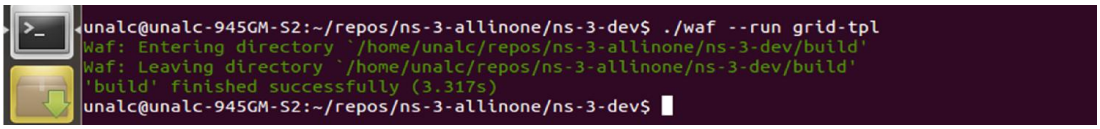
Şekil 6.11. Bilgisayar 1'de NS-2 ızgara topoloji simülasyonu sistem durumu



Şekil 6.12. Bilgisayar 2'de NS-2 ızgara topoloji simülasyonu sistem durumu

6.1.2. NS-3 ağ simülatöründe ızgara topoloji simülasyonu

Izgara topolojinin NS-3 ağ simülatörü üzerinde gerçeklenmesi için kullanılan kod EK-4'te verilmiştir. NS-3 ağ simülatöründe, simülasyonun çalıştırılması için C++ dilinde hazırlanmış olan grid-tpl isimli simülasyon dosyasının bulunduğu konuma gelinerek, run komutu ile simülasyon gerçekleştirilebilir. Şekil 6.13'de gerçekleştirilen işleme ait ekran çıktısı görülmektedir.

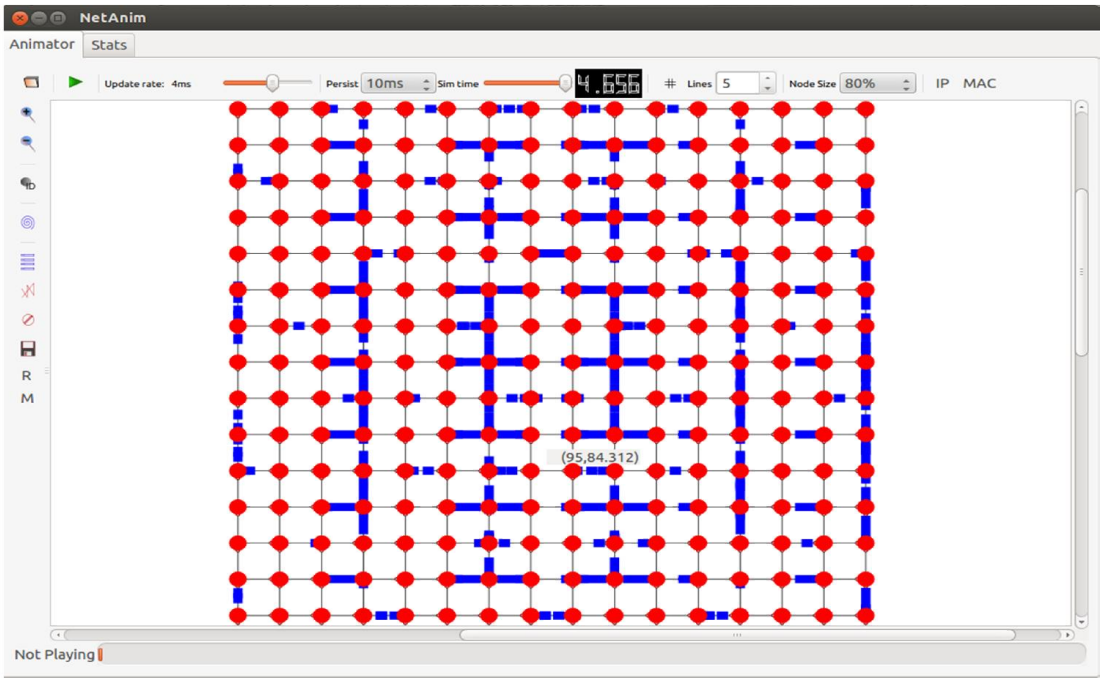


Şekil 6.13. Terminalde NS-3 simülatörünü çalıştırma komutları

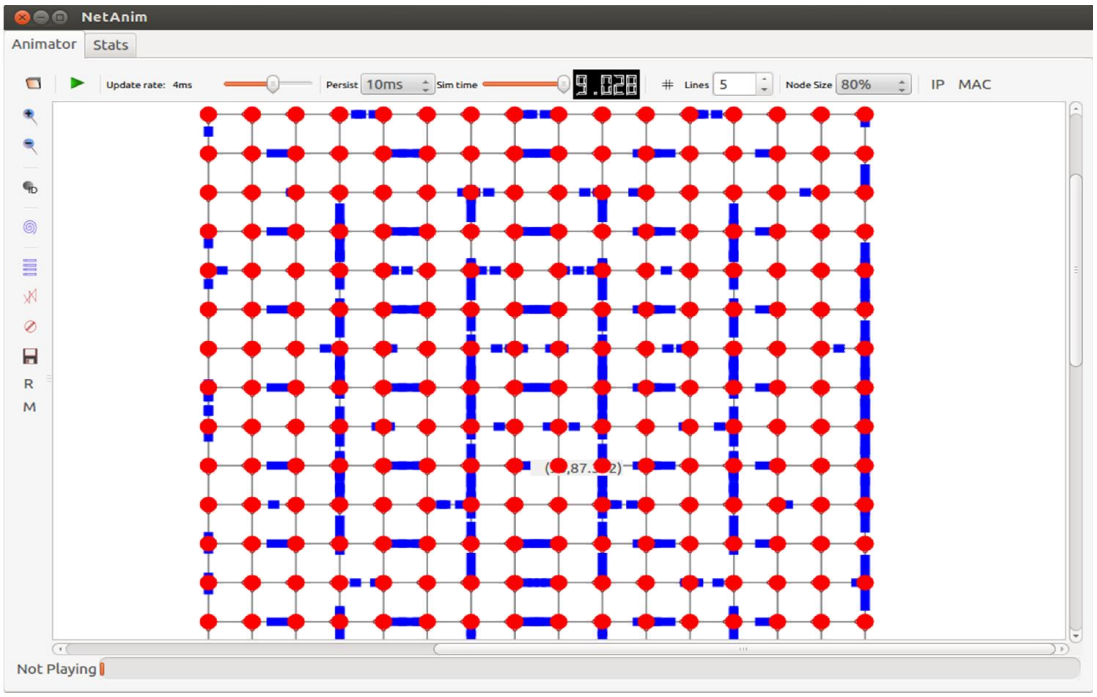
6.1.2.1. NS-3 ızgara topoloji simülasyon çıktıları

NS-3' te simülasyonun çalışmasını görebilmek için programın üretmiş olduğu grid-animation.xml dosyası NetAnim programı kullanılıp açılarak, simülasyonun görselleştirilmesi sağlanmaktadır. Şekil 6.14 ve 6.15'de NS-3 ağ simülatörü üzerinde

gerçekleştirilen ızgara ağ simülasyonuna ait, NetAnim programı kullanılarak elde edilen simülasyon çıktıları görülmektedir. Şekil 6.14'de simülasyon gerçekleşmesi sırasında bilgisayar1 üzerindeki görüntü ve Şekil 6.15'de simülasyonun gerçekleşmesi sırasında bilgisayar2'de kaydedilen görüntü bulunmaktadır.



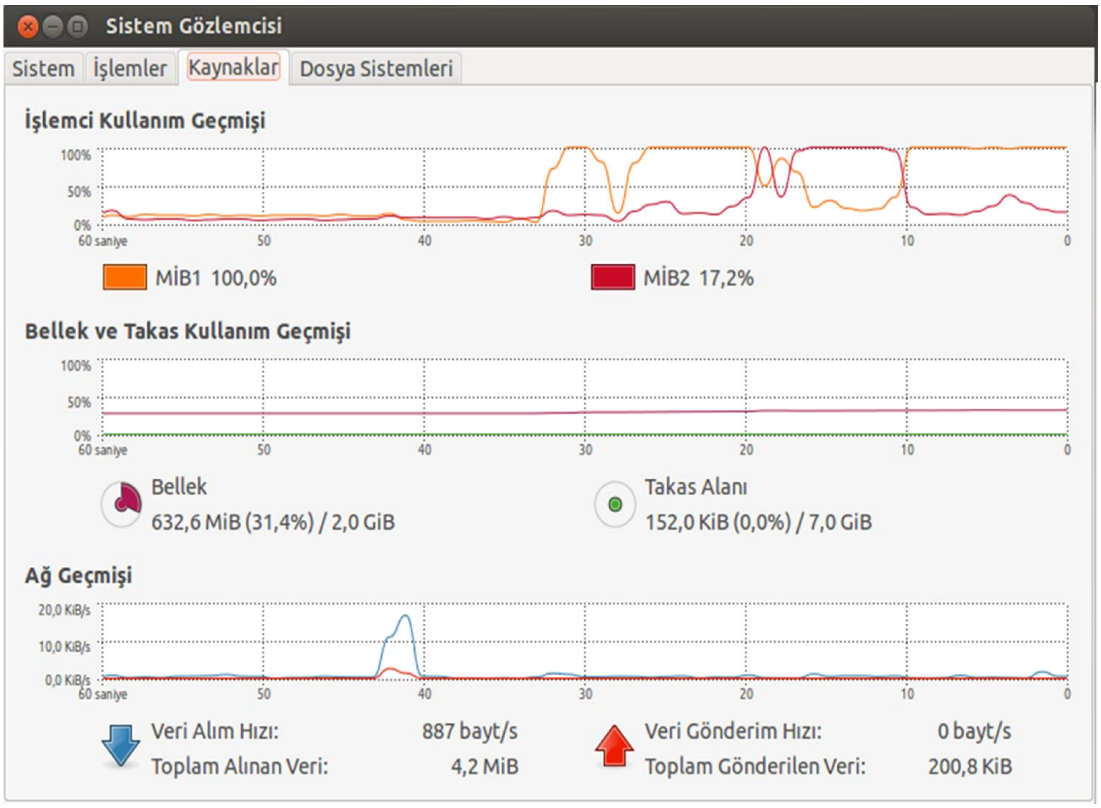
Şekil 6.14. Bilgisayar 1'de NS-3 ızgara simülasyonunun 4.6 sn'deki trafik durumu



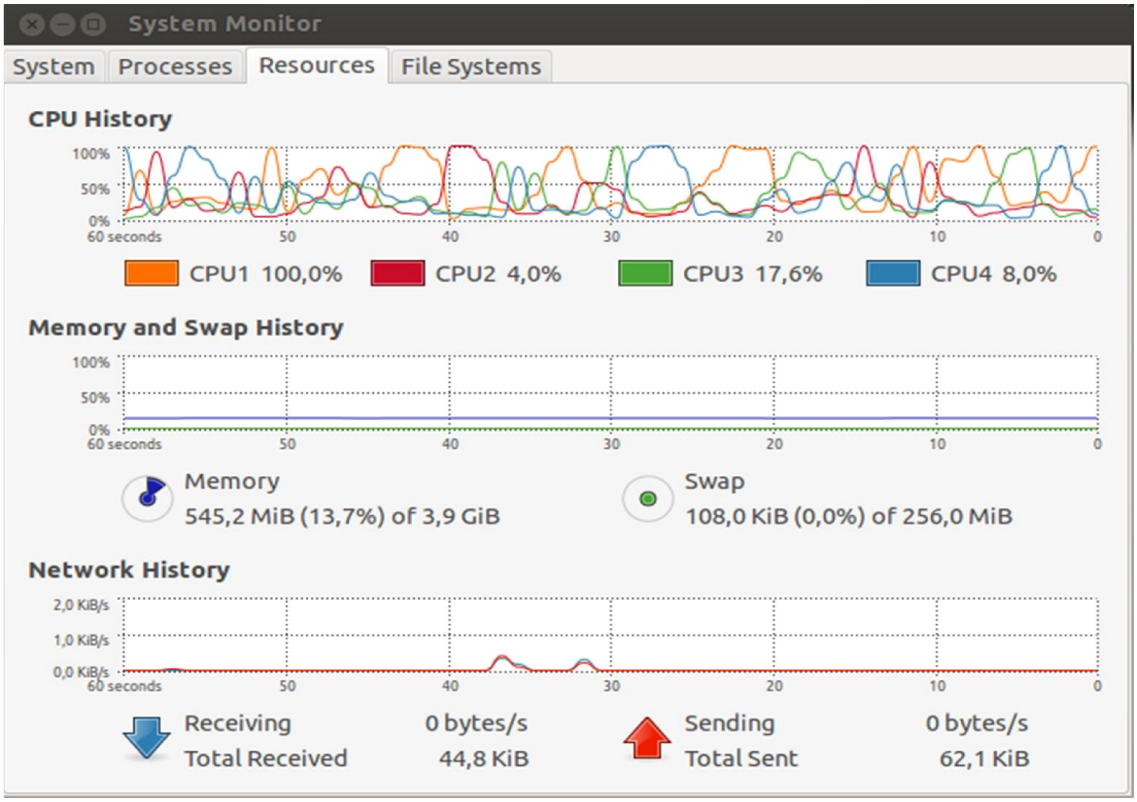
Şekil 6.15. Bilgisayar 2'de NS-3 ızgara simülasyonunun 9.sn'deki trafik durumu

6.1.2.2. NS-3 ağ simülatöründe cpu ve belek kullanımı

Şekil 6.16, 6.17'de NS-3 ağ simülatörü üzerinde ızgara simülasyonun çalıştırılması sırasında bilgisayar1 ve bilgisayar2'deki işlemci ve bellek kullanımı gösterilmiştir. Bu işlemlerin gerçekleştirilmesi sırasında Ubuntu sistem gözlemcisinden yararlanılmıştır. Bilgisayar1 ve bilgisayar2'de farklı işlemci sayıları mevcuttur. Grafikte simülasyon süresince işlemcilere ait kullanım yüzdeleri, bellek kullanım miktarları, ağ üzerindeki trafik miktarına ait bilgiler görülmektedir. Bilgisayar1 ve bilgisayar2'de işlem zamanı boyunca 1 adet işlemcinin %100 oranında kullanıldığı tespit edilmiştir. Fakat işlem süresince simülasyon işlemi işletim sistemi tarafından farklı işlemciler üzerine tahsis edilmiştir.



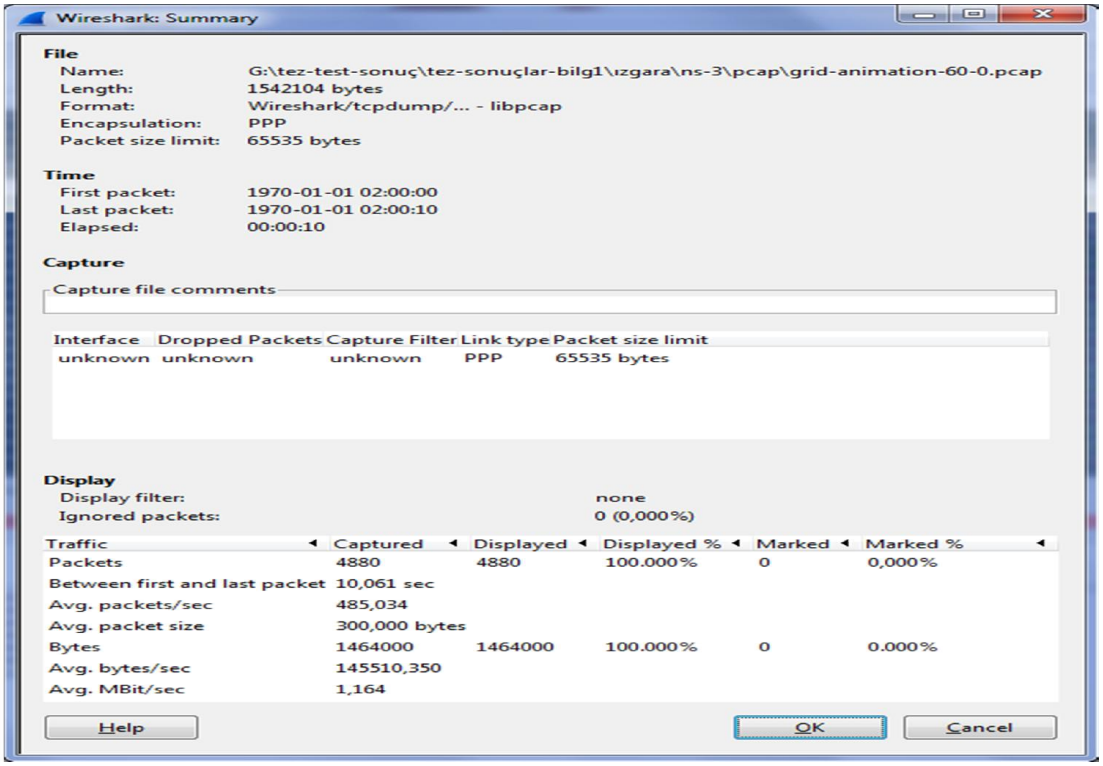
Şekil 6.16. Bilgisayar 1'de NS-3 ızgara topoloji simülasyon sistem durumu



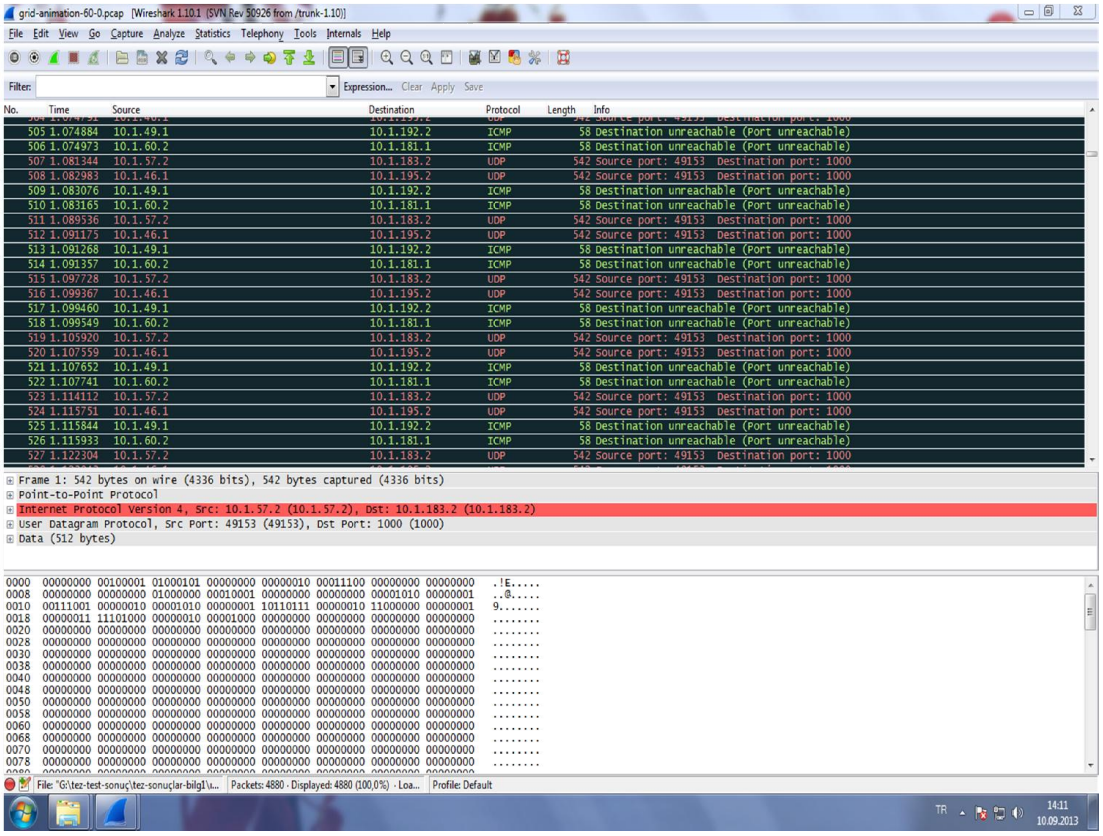
Şekil 6.17. Bilgisayar 2'de NS-3 ızgara topoloji simülasyon sistem durumu

6.1.2.3. NS-3 ağ simülatörü ızgara topoloji ağ çıkış değerleri

NS-3 ağ simülatörü üzerinde gerçekleştirilen ızgara ağ simülasyonu sonucu, düğümler arasında gerçekleşen trafiği irdelemek için ağ simülatörünün üretmiş olduğu pcap dosyalarını wireshark programını kullanarak açabiliriz. Pcap dosyaları açıldığında simülasyon süresi boyunca gerçekleşen trafik akışına ait olan detaylı paket bilgileri karşımıza çıkacaktır.



Şekil 6.18. NS-3 ızgara topoloji 0-60 düğümleri arasındaki trafik özeti

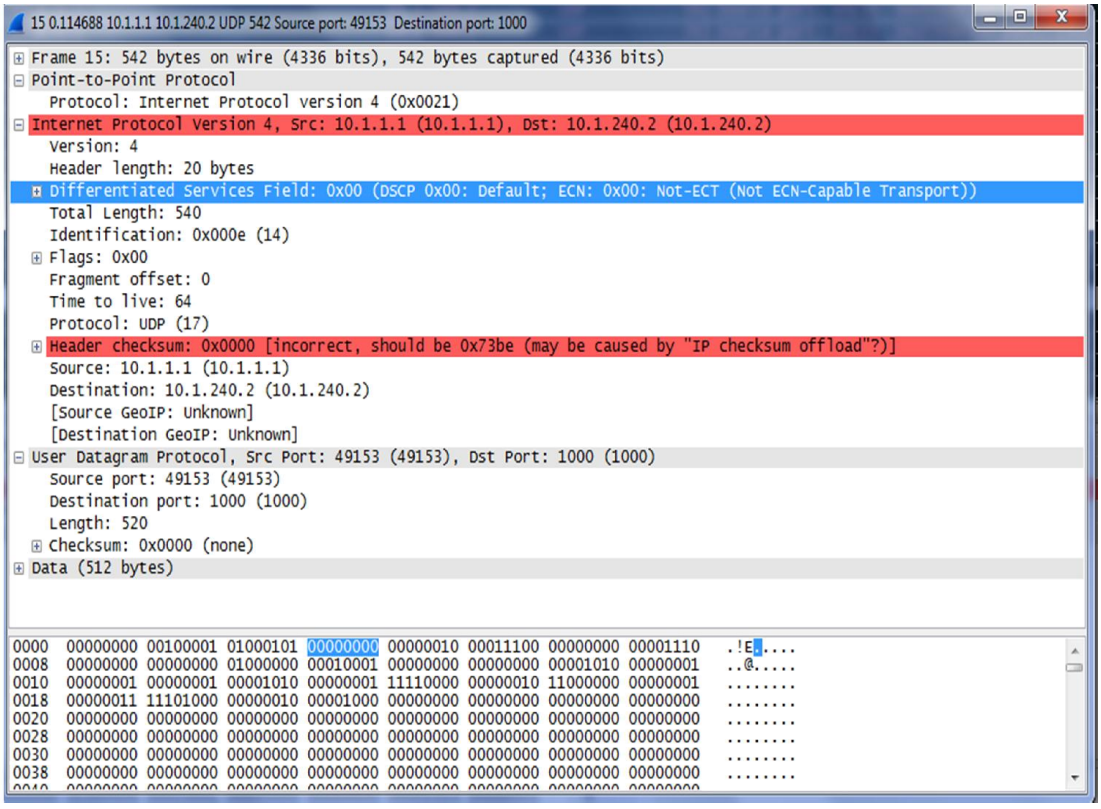


Şekil 6.19. NS-3 ızgara topolojisinde 0-60 düğümleri arası paketler

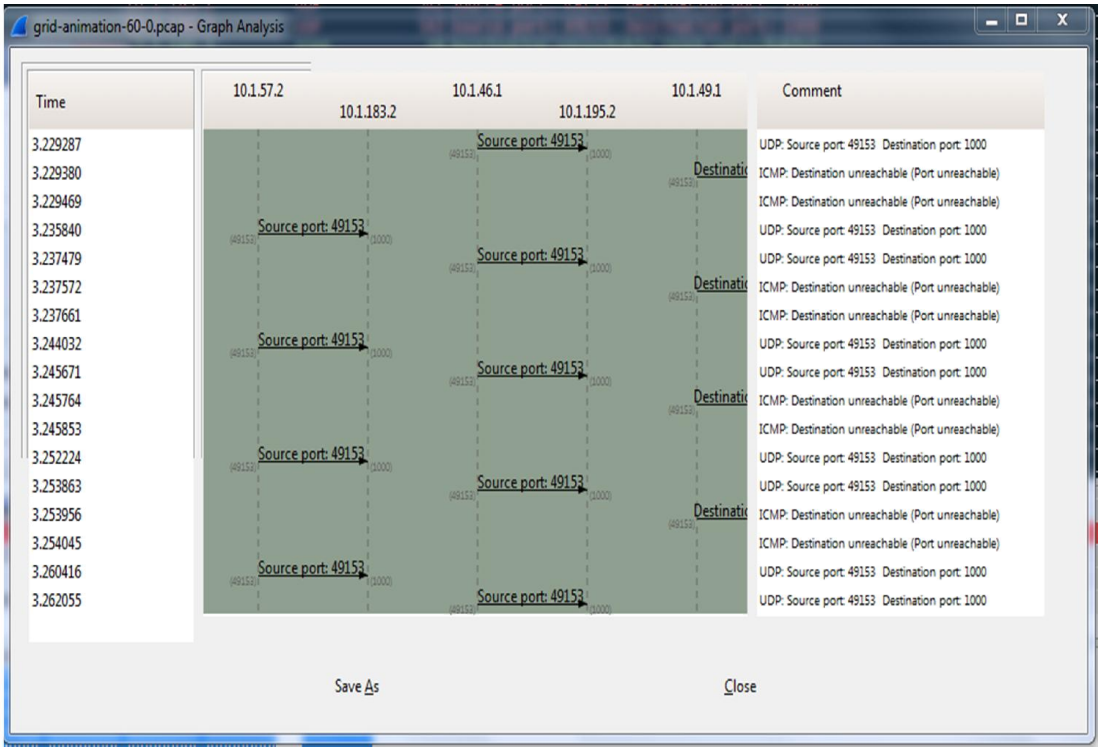
Şekil 6.18'de 0-60 düğümleri arasında gerçekleşen trafiğe ait özet bir bilgi görülmektedir. Simülasyon süresi yakalanan ve gösterilen paket bilgileri ortalama paket boyutu ve ortalama paket adeti bilgileri ve daha bir çok bilgi sunulmaktadır.

Şekil 6.19'de 0-60 düğümleri arasında gerçekleşen trafiğe ait yakalanan çerçevelerin listesi gösterilmektedir. Burada incelenmek istenilen çerçeve bilgisi tıklanarak detaylı içeriğe ulaşılabilir. Genel görünümde ise, çerçeve no, çerçevenin yakalanma zamanı, kaynak ve hedef IP adres bilgileri, protokol bilgileri, çerçeve uzunluğu ve bilgi kısmı bulunmaktadır.

Şekil 6.20'de 0-60 düğümleri arasında gerçekleşen trafiğe ait paketler içerisinden 15 nolu çerçeveye ait detaylı içerik görüntülenmektedir. Bu çerçeve içeriğinde çerçeve boyutu IP protokol bilgileri, taşıma katmanı protokolü UDP'ye ait detaylı bilgi ve taşınmakta olan veri ile ilgili bilgiler bulunmaktadır. Bit bazında bu bilgileri görüntülemek ve incelemek mümkündür.



Şekil 6.20. NS-3 ızgara topolojisinde bir pakete ait içerik



Şekil 6.21. NS-3 ızgara topolojisinde 0-60 düğümleri arası akış diagramı

Şekil 6.21'de 0-60 düğümleri arasında gerçekleşen trafiğe ait akış diagramı görülmektedir. Zamana bağlı olarak, hangi düğümler üzerinden akışın gerçekleştiği, IP, port numaraları ve yorum bölümü grafiksel olarak yer almaktadır.

6.1.3. Izgara ağ modeli için değerlendirme

Tablo 6.3 ve 6.4'te Bilgisayar1 ve Bilgisayar2 üzerinde NS-2 ve NS-3 ağ simülatörleri üzerinde gerçekleştirilen ızgara simülasyonuna ait test sonuçları görülmektedir. Simülasyonların gerçekleştirilmesi ile farklı donanımlar ve simülatörler üzerinde simülasyon süreleri ve simülatörlerin bilgisayar kaynaklarını (işlemci ve bellek) ne kadar kullandığı tespit edilmiştir.

Tablo 6.3. Bilgisayar1 Izgara Simülasyon Sonuçları

SİMÜLATÖR	SİMÜLASYON SÜRESİ	KAYNAK KULLANIMI			DÜĞÜM SAYISI
		CPU		BELLEK	
		1	2		
NS-2	75 sn.	%19	%100	%25	256
NS-3	310 sn.	%15	%100	%33	256

Tablo 6.4. Bilgisayar2 Izgara Simülasyon Sonuçları

SİMÜLATÖR	SİMÜLASYON SÜRESİ	KAYNAK KULLANIMI					DÜĞÜM SAYISI
		CPU				BELLEK	
		1	2	3	4		
NS-2	46 sn.	%100	%20	%17	% 05	%9	256
NS-3	190 sn.	%100	%04	%18	%08	%15	256

Simülatörler için ızgara ağ modeli örneklerini inceleyecek olursak, 2. bilgisayarın özellikleri 1. bilgisayara göre daha iyi olduğu için, 2.bilgisayarda NS-2 ve NS-3 uygulaması daha kısa zamanda gerçekleşmiştir. Ancak aynı koşul altında, tek bir bilgisayarda değerlendirecek olursak, her iki bilgisayarda da simülasyon NS-3'te daha uzun zamanda gerçekleşmiştir. NS-2 ve NS-3'de simülasyon çalışırken birbirine yakın yüzdelerde işlemci kullanımı olmuştur. Her iki bilgisayarda da işlemcinin %99'u simülasyon çalışırken kullanılmıştır. Bellek kullanımı NS-3' te daha yüksek olduğu gözlenmiştir. Bu bilgilerin tümü değerlendirildiğinde iki simülatör arasında performans farklılığının olduğu açıkça görülmektedir. NS-3 ağ simülatöründe simülasyon süresi daha uzun sürmesine rağmen, donanımsal yapıları daha etkili bir şekilde kullandığı görülmektedir.

6.2. Yıldız Topoloji

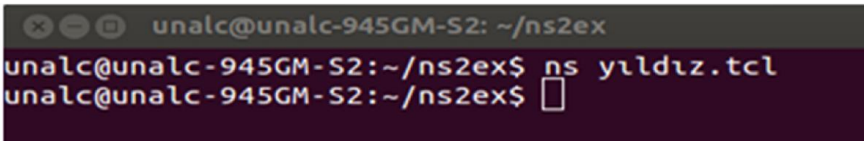
Bu bölümde yıldız topoloji modeli, NS-2 ve NS-3 simülatörleri kullanılarak, iki farklı bilgisayarda çalıştırılmış ve performans analizleri yapılmıştır. Yıldız ağ modelinde kullanılan parametreler Tablo 6.5'de verilmiştir. Ağ simülasyon parametresi olarak kullanılan protokol, bant genişliği, gecikme zamanı, simülasyon süresi ve düğüm sayısı değerleri bulunmaktadır.

Tablo 6.5. Yıldız topoloji simülasyonu parametreleri

PARAMETRE	DEĞER
Protokol	TCP
Bant Genişliği	5 mbps
Gecikme	2 ms.
Simülasyon Süresi	10 sn
Düğüm Sayısı	500
Paket boyutu	150 mb.
Veri hızı	15 kb/s
Kuyruk yapısı	Drop-tail

6.2.1. NS-2 Ağ Simülatöründe Yıldız Topoloji Simülasyonu

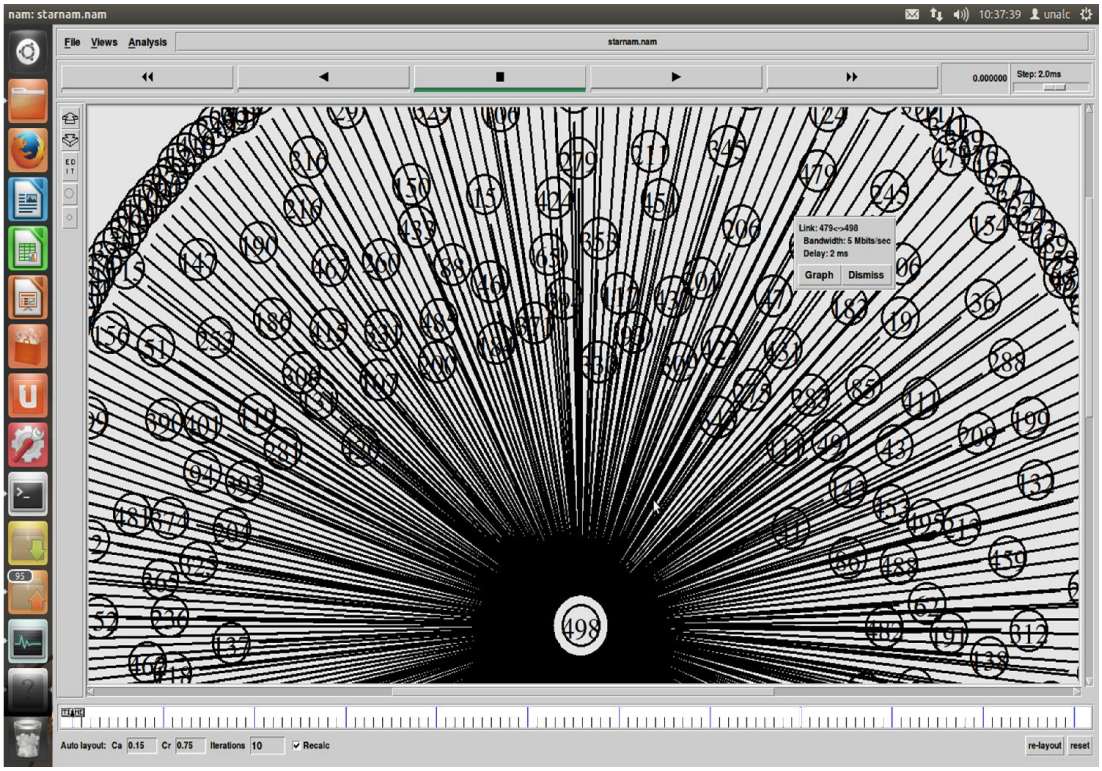
Yıldız topolojinin NS-2 ağ simülatörü üzerinde gerçeklenmesi için kullanılan kod EK-5'te verilmiştir. NS-2 ağ simülatöründe, simülasyonun çalıştırılması için hazırlanmış olan yıldız.tcl isimli simülasyon dosyasının bulunduğu konuma gelinerek, ns komutu ile simülasyon gerçekleştirilebilir. Şekil 6.22'de gerçekleştirilen işleme ait ekran çıktısı görülmektedir.



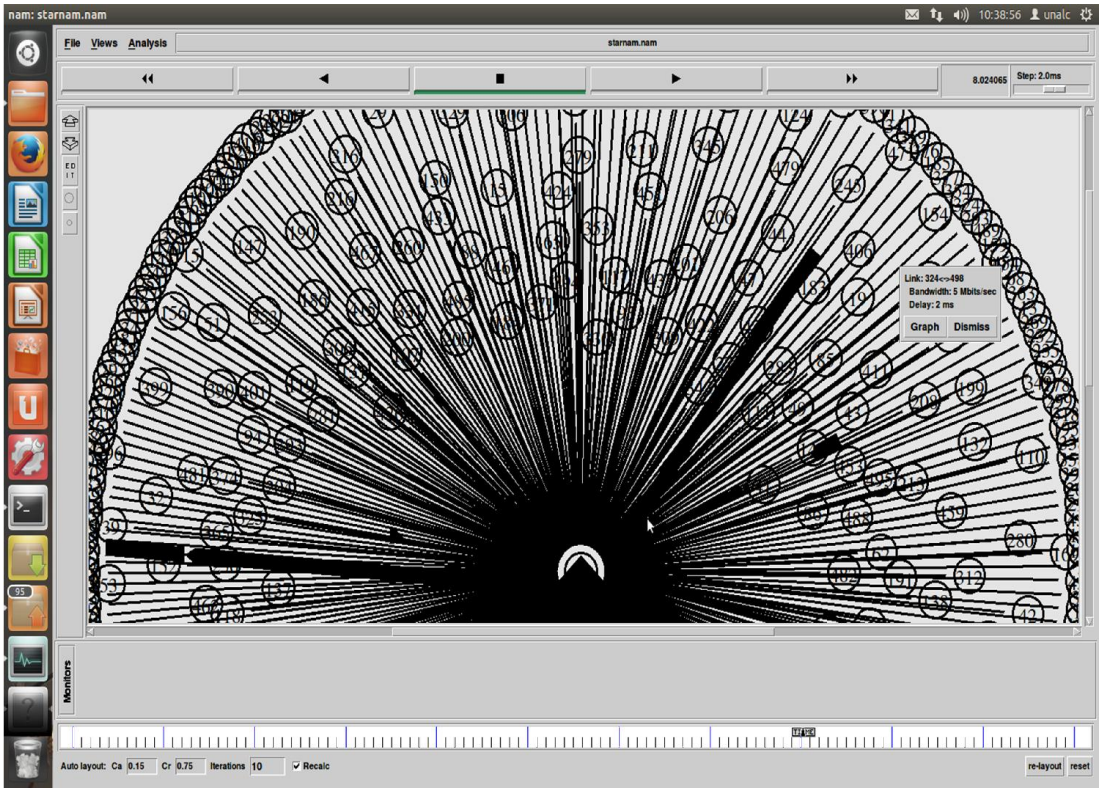
Şekil 6.22. Star toplojinin NS-2 'de çalıştırılması-konsol ekranı

6.2.1.1. NS-2 yıldız topoloji simülasyon çıktıları

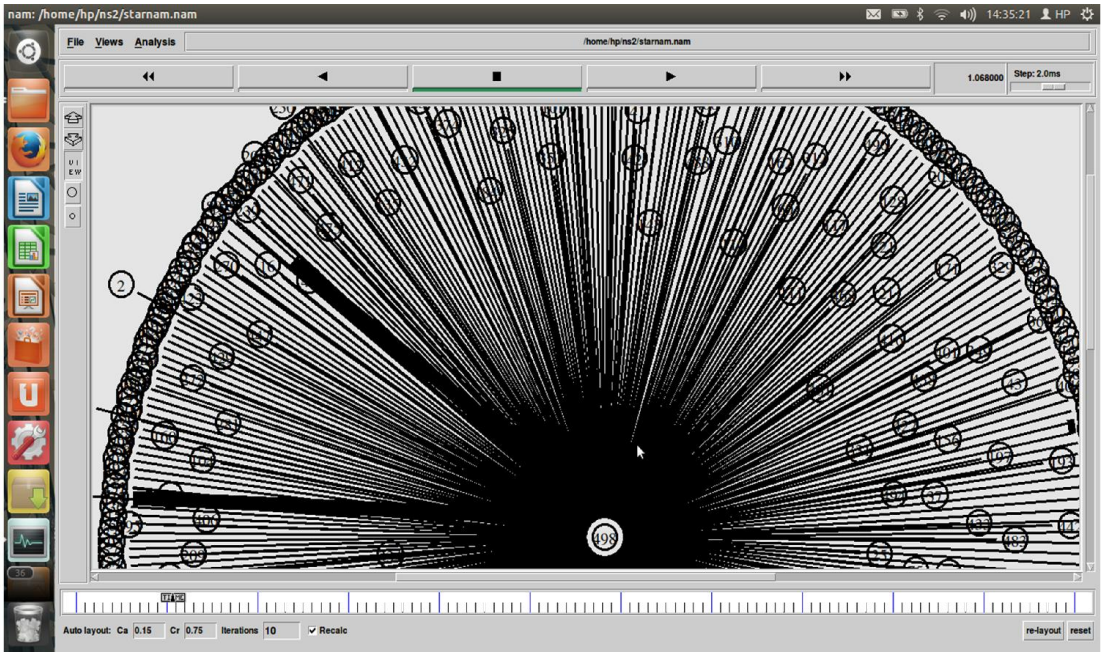
Şekil 6.23, 6.24 ve 6.25'de NS-2 ağ simülatörü üzerinde gerçekleştirilen yıldız ağ simülasyonuna ait, network animatör (NAM) programı kullanılarak elde edilen simülasyon çıktıları görülmektedir. Şekil 6.23 ve 6.24'de simülasyonun gerçekleşmesi sırasında bilgisayar1' de, Şekil 6.25'de bilgisayar2'de kaydedilen görüntü bulunmaktadır.



Şekil 6.23. Bilgisayar 1'de NS-2 star topoloji simülasyonu



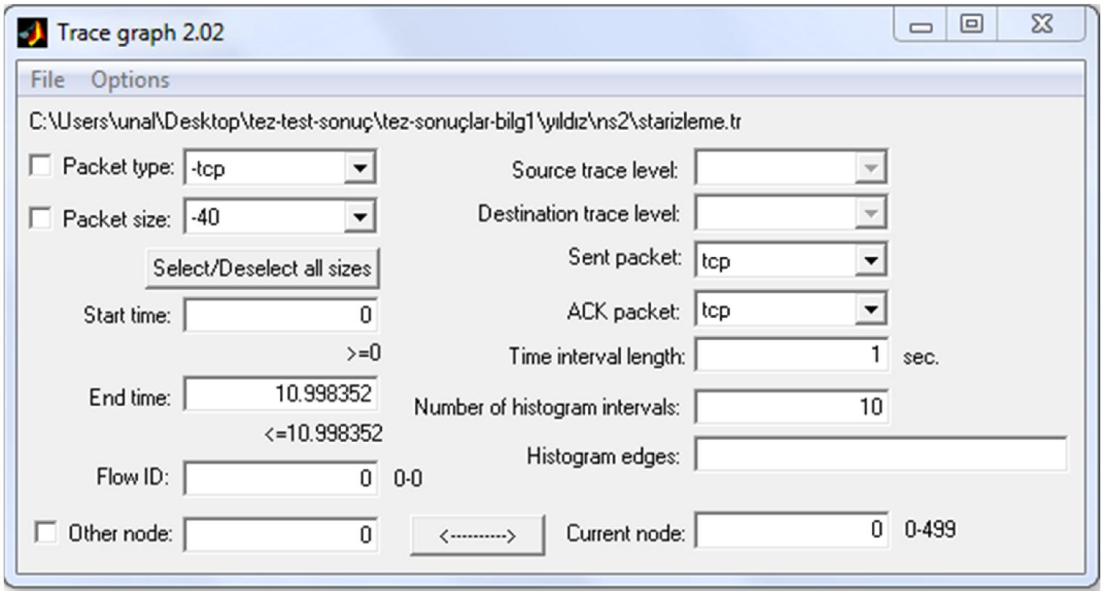
Şekil 6.24. Bilgisayar 1'de NS-2 star topoloji simülasyonu



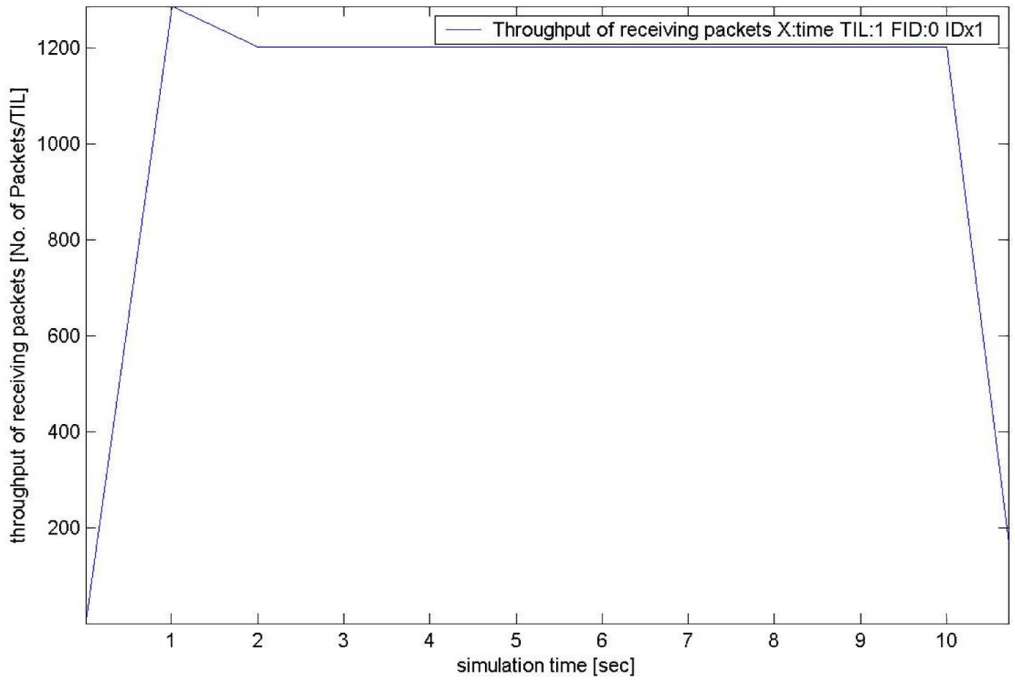
Şekil 6.25. Bilgisayar 2'de NS-2 star topoloji simülasyonu

6.2.1.2. NS-2 yıldız topoloji ağ çıkış değerleri

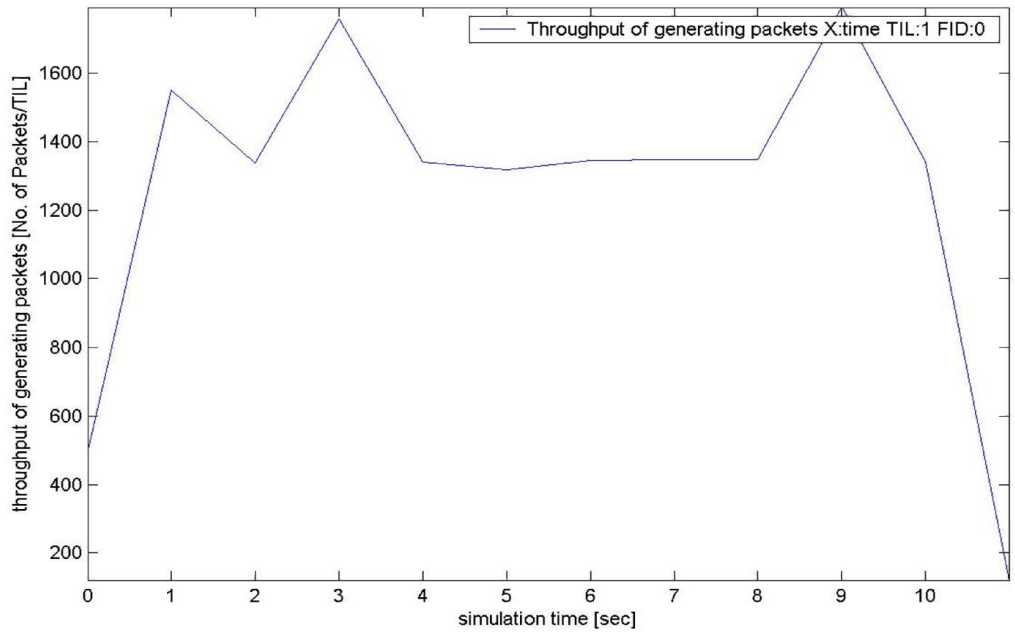
Yıldız ağ simülasyonun NS-2 ağ simülatörü üzerinde çalıştırılması sonucu elde edilen tr uzantılı iz dosyasının tracegraph programına yüklenmesiyle ağ simülasyonu ile ilgili birçok istatiksel sonuçlar elde edilebilmektedir. Şekil 6.26'da tracegraph programına ait ana ekran görüntüsü görülmektedir.



Şekil 6.26. Trace graph programına ait ana ekranı



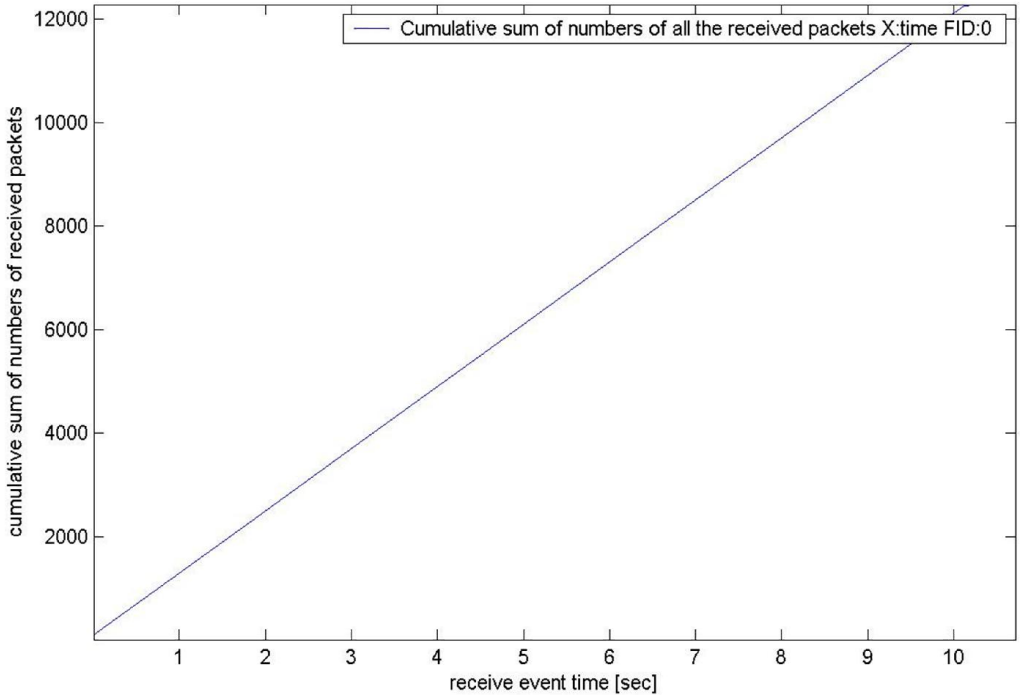
Şekil 6.27. Simülasyon zamanı ve alınan paketlerin ağ çıkış değerleri



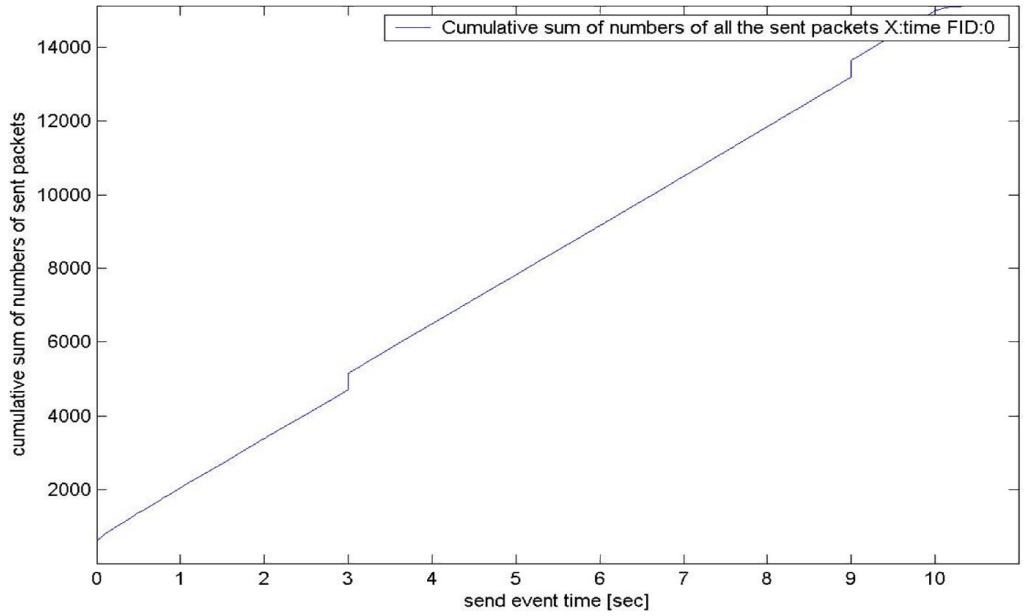
Şekil 6.28. Simülasyon zamanı ve üretilen paketlerin ağ çıkış değerleri

Şekil 6.27'de simülasyon zamanına bağlı olarak, alınan paketlerin ağ çıkış değerlerine ait grafik görülmektedir. Grafikte 1 ve 2. sn'de yükselen bir grafik seyreden değer 1. sn'de max. değere ulaşmış ve 1-10 sn arası bu şekilde devam etmiştir. Şekil 6.28'de simülasyon zamanı boyunca üretilen paketlerin ağ çıkış değerlerine ait grafik görülmektedir. Bu grafik zamana bağlı olarak dalgalı bir seyir izlemiştir. Simülasyonun sonlanması ile 0 değerini almıştır.

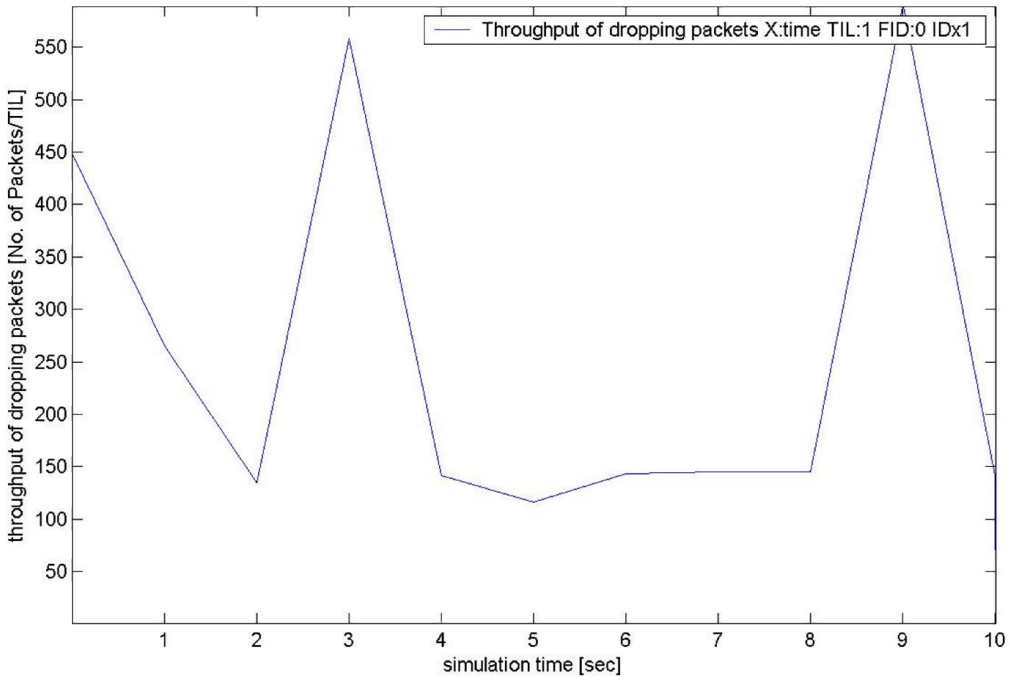
Şekil 6.29'da simülasyon zamanı boyunca paket alımının zamana bağlı değişimi ve alınan toplam paket sayılarına ait grafik görülmektedir. Simülasyon süresi boyunca alınan toplam paket sayısı doğrusal olarak artış göstermiştir. Şekil 6.30'da simülasyon zamanı boyunca paket alımının zamana bağlı değişimi ve gönderilen toplam paket sayılarına ait grafik görülmektedir. Simülasyon süresi boyunca gönderilen toplam paket sayısı doğrusal olarak artış göstermiştir.



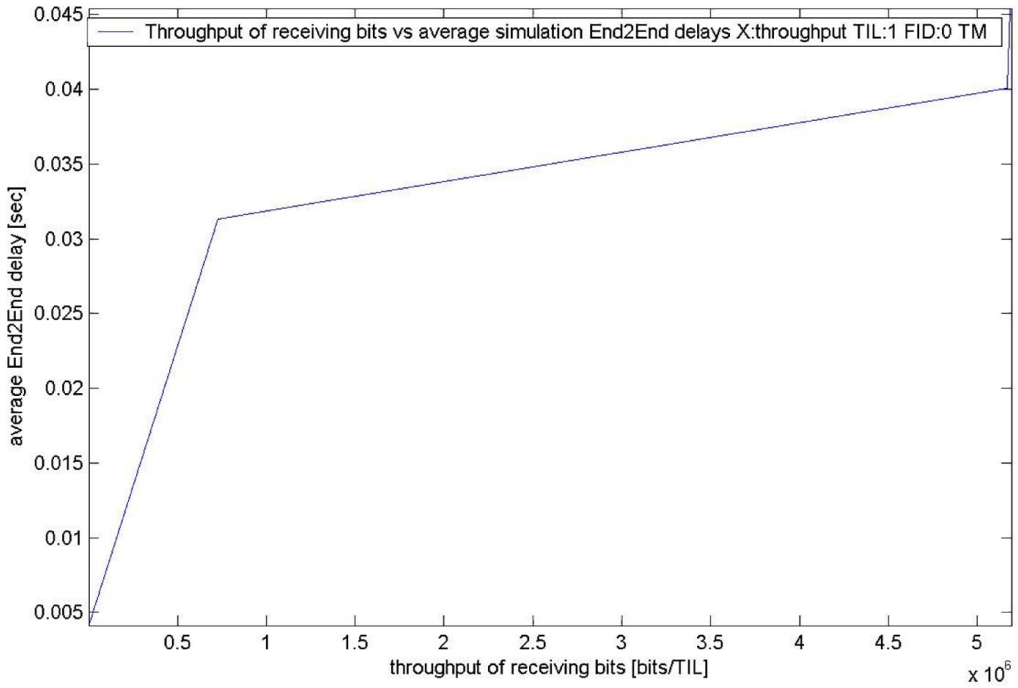
Şekil 6.29. Paketlerin alım zamanı ve alınan paketlerin toplamı



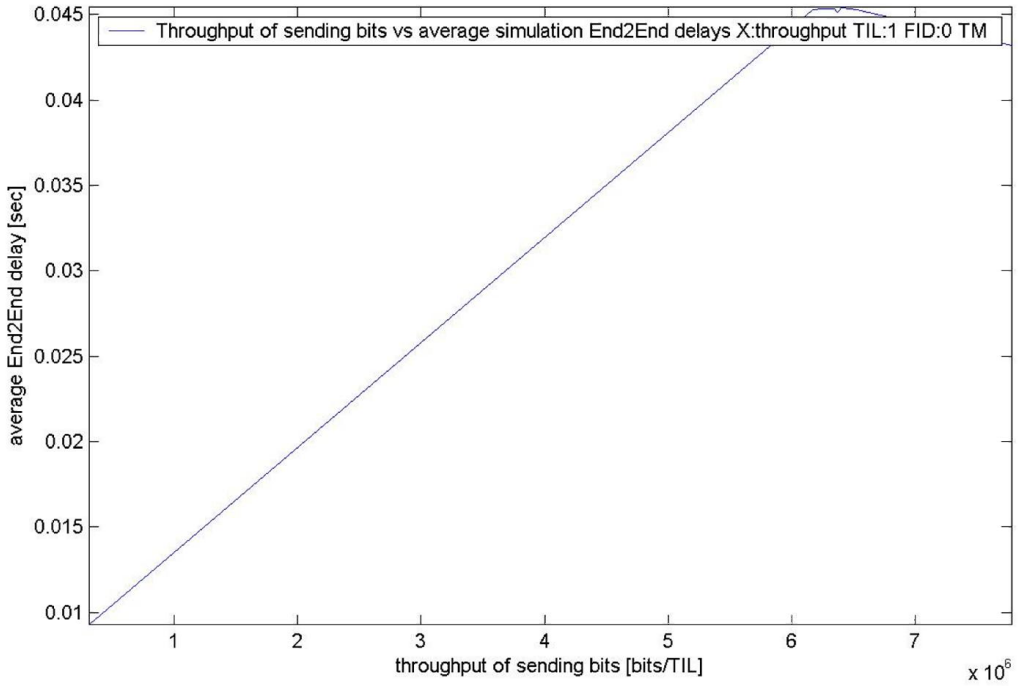
Şekil 6.30. Paketlerin gönderim zamanı ve gödnerilen paketlerin toplamı



Şekil 6.31. Simülasyon zamanı ve düşürülen paketlerin ağ çıkış değerleri



Şekil 6.32. Alınan bitlerin ağ çıkış değeri ve ortalama gecikme zamanı

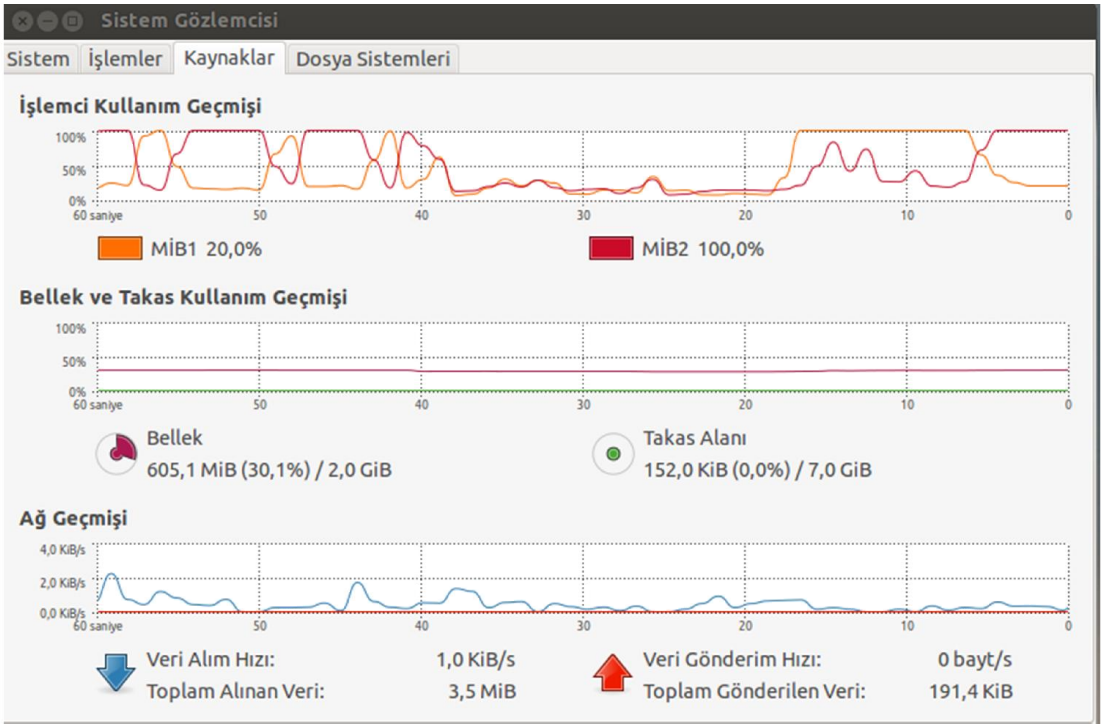


Şekil 6.33. Gönderilen bitlerin ağ çıkış değeri ve ortalama gecikme zamanı

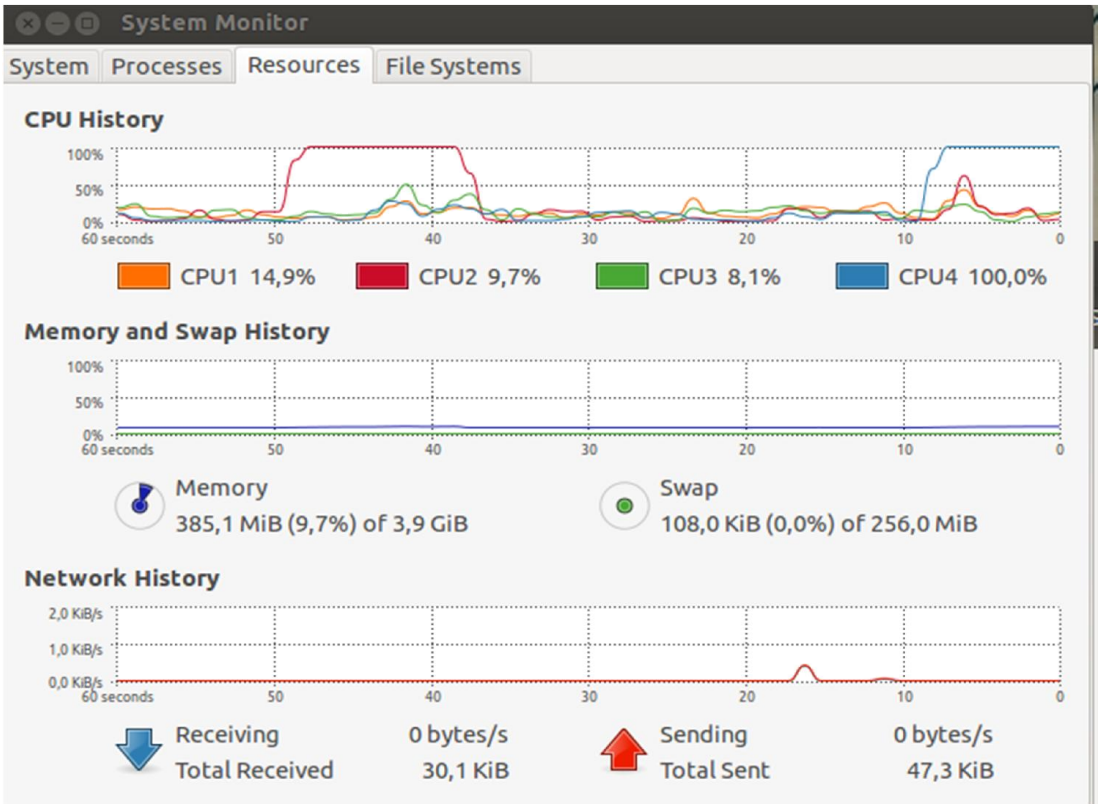
Şekil 6.31’ de simülasyon zamanına bağlı olarak, düşürülen paketlere ait ağ çıkış grafiği görülmektedir. Düşürülen paket ağ çıkış trafiği simülasyon süresi boyunca 100-600 arasında değişik değerler almıştır. Şekil 6.32 ve 6.33’de simülasyon zamanı boyunca alınan ve gönderilen bitlere ait ağ çıkış değerlerinin noktadan noktaya ortalama gecikme zamanı değerleri görülmektedir. Alınan bitlerin ağ çıkış değerlerine ait ortalama gecikme azalış gösterirken, gönderilen bitlerin ağ çıkış değerlerine ait ortalama gecikme zamanı doğrusal bir artış göstermiştir.

6.2.1.3. NS-2 ağ simülatöründe cpu ve bellek kullanımı

Şekil 6.34 ve 6.35’te, NS-2 simülatörü üzerinde yıldız simülasyonun çalıştırılması sırasında bilgisayar1 ve bilgisayar2’deki işlemci ve bellek kullanımı gösterilmiştir. Bu işlemlerin gerçekleştirilmesi sırasında Ubuntu sistem gözlemcisinden yararlanılmıştır. Bilgisayar1 ve bilgisayar2’de farklı işlemci sayıları mevcuttur. Grafikte simülasyon süresince işlemcilere ait kullanım yüzdeleri, bellek kullanım miktarları, ağ üzerindeki trafik miktarına ait bilgiler görülmektedir. Bilgisayar1 ve bilgisayar2’de işlem zamanı boyunca 1 adet işlemcinin %100 oranında kullanıldığı tespit edilmiştir. Fakat işlem süresince simülasyon işlemi işletim sistemi tarafından farklı işlemciler üzerine tahsis edilmiştir. NS-2 ağ simülatörü üzerinde, yıldız topoloji simülasyonu oldukça kısa sürede tamamlanmıştır.



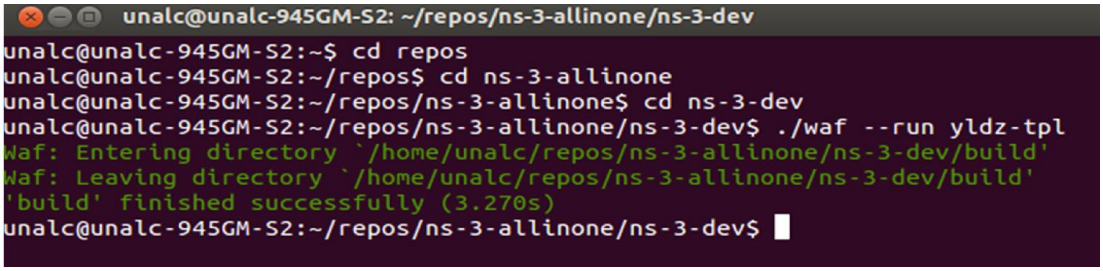
Şekil 6.34. Bilgisayar 1'de NS-2 Yıldız topoloji simülasyonu sistem durumu



Şekil 6.35. Bilgisayar 2'de NS-2 Yıldız topoloji simülasyonu sistem durumu

6.2.2. NS-3 ağ simülatöründe yıldız topoloji simülasyonu

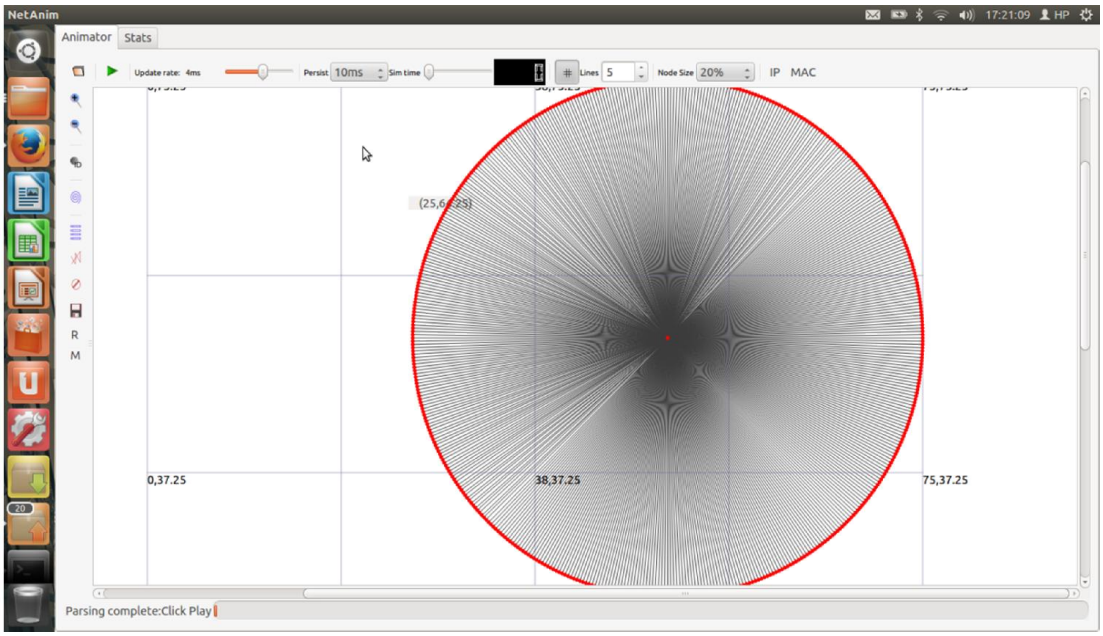
Yıldız topolojinin NS-3 ağ simülatörü üzerinde gerçeklenmesi için kullanılan kod EK-6'da verilmiştir. NS-3 ağ simülatöründe, simülasyonun çalıştırılması için C++ dilinde hazırlanmış olan yldz-tpl isimli simülasyon dosyasının bulunduğu konuma gelinerek, run komutu ile simülasyon gerçekleştirilebilir. Şekil 6.36'da gerçekleştirilen işleme ait ekran çıktısı görülmektedir.



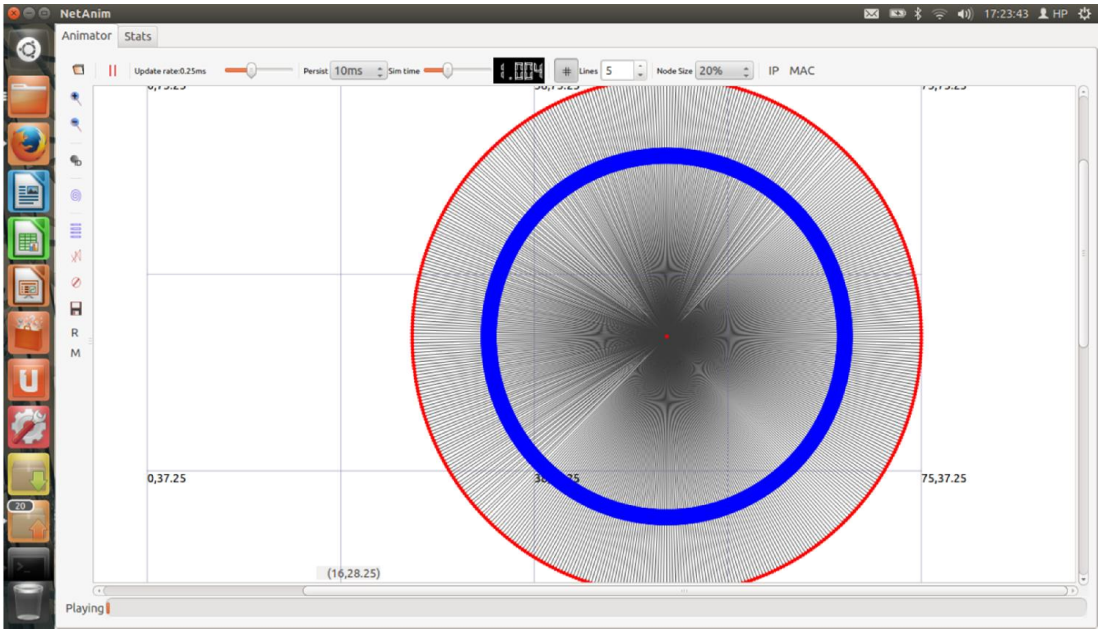
Şekil 6.36. NS-3'de Yıldız topoloji kodlarının çalıştırılması-konsol ekranı

6.2.2.1. NS-3 yıldız topoloji simülasyon çıktıları

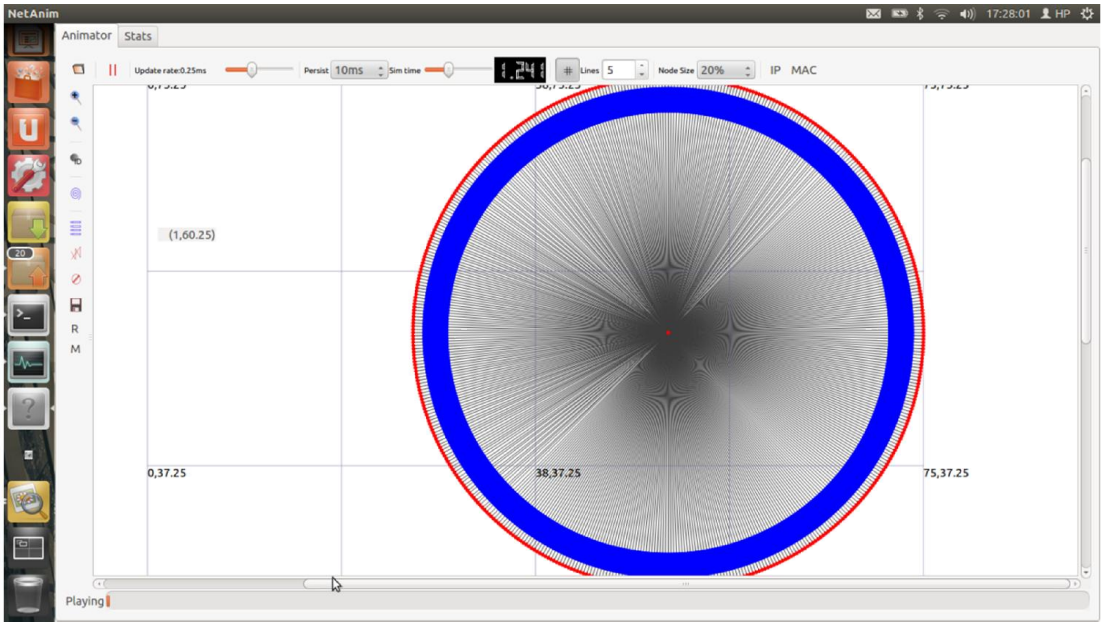
Şekil 6.37, 6.38 ve 6.39'da NS-3 ağ simülatörü üzerinde gerçekleştirilen yıldız ağ simülasyonuna ait, NetAnim programı kullanılarak elde edilen simülasyon çıktıları görülmektedir. Şekil 6.37'de simülasyon başlamadan önce, bilgisayar1 üzerinde oluşan görüntü ve Şekil 6.38 ve 6.39'da simülasyonun gerçekleşmesi sırasında bilgisayar2 üzerinde kaydedilen görüntü bulunmaktadır.



Şekil 6.37. Bilgisayar 1'de NS-3 Yıldız topoloji görünümü



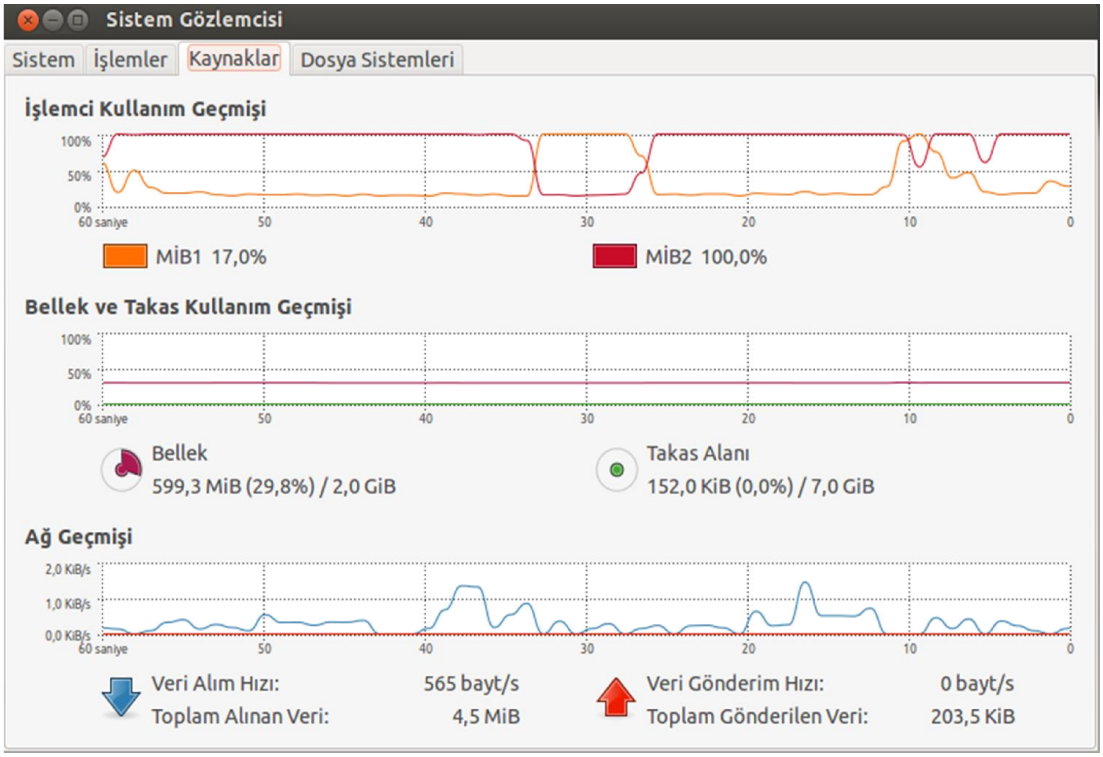
Şekil 6.38. Bilgisayar 2'de NS-3 Yıldız topoloji görünümü



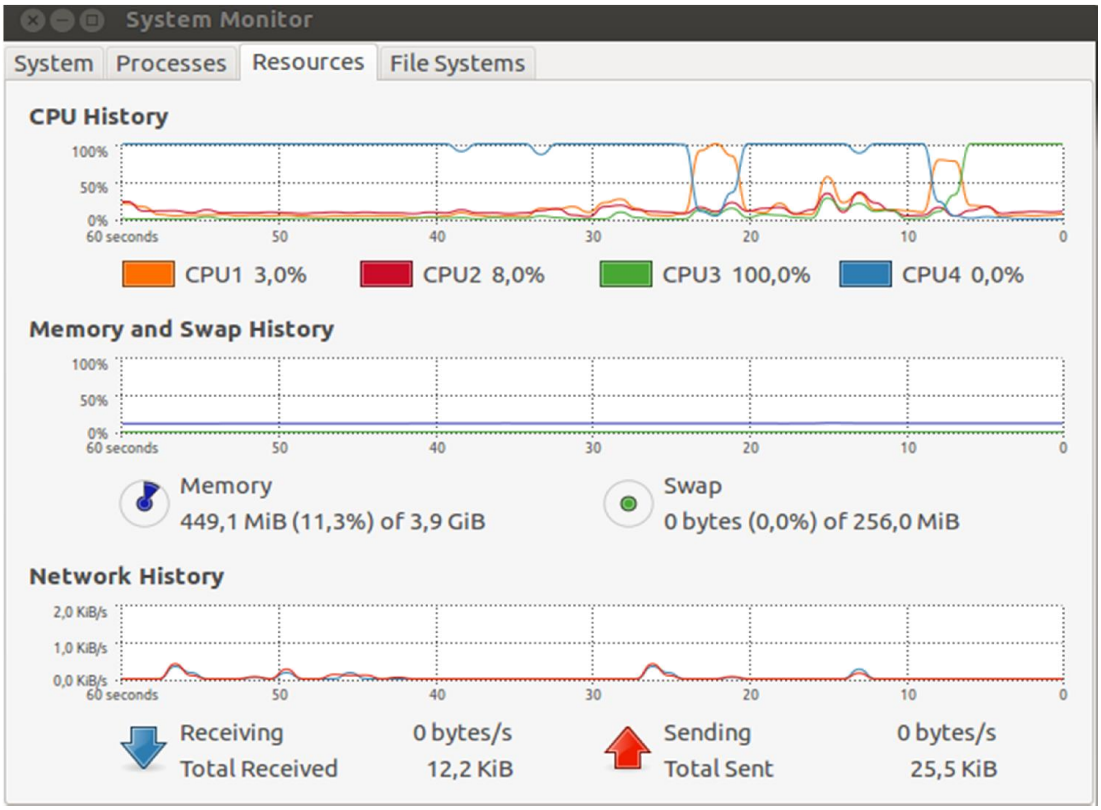
Şekil 6.39. Bilgisayar 2'de NS-3 Yıldız topoloji görünümü

6.2.2.2. NS-3 ağ simülatöründe cpu ve belek kullanımı

Şekil 6.40 ve 6.41'de, NS-3 simülatörü üzerinde yıldız simülasyonun çalıştırılmadan önce ve çalıştırılması sırasında bilgisayar1 ve bilgisayar2'deki işlemci ve bellek kullanımı gösterilmiştir. Bu işlemlerin gerçekleştirilmesi sırasında Ubuntu sistem gözlemcisinden yararlanılmıştır. Bilgisayar1 ve bilgisayar2 'de farklı işlemci sayıları mevcuttur. Grafikte simülasyon süresince işlemcilere ait kullanım yüzdeleri, bellek kullanım miktarları, ağ üzerindeki trafik miktarına ait bilgiler görülmektedir. Bilgisayar1 ve bilgisayar2'de işlem zamanı boyunca 1 adet işlemcinin %100 oranında kullanıldığı tespit edilmiştir. Fakat işlem süresince simülasyon işlemi işletim sistemi tarafından farklı işlemciler üzerine tahsis edilmiştir. NS-3 ağ simülatörü üzerinde yıldız topoloji simülasyonu oldukça kısa sürede tamamlanmıştır. NS-3 ağ simülatöründe bellek kullanımının, NS-2 ağ simülatörüne göre yüksek olduğu tespit edilmiştir.



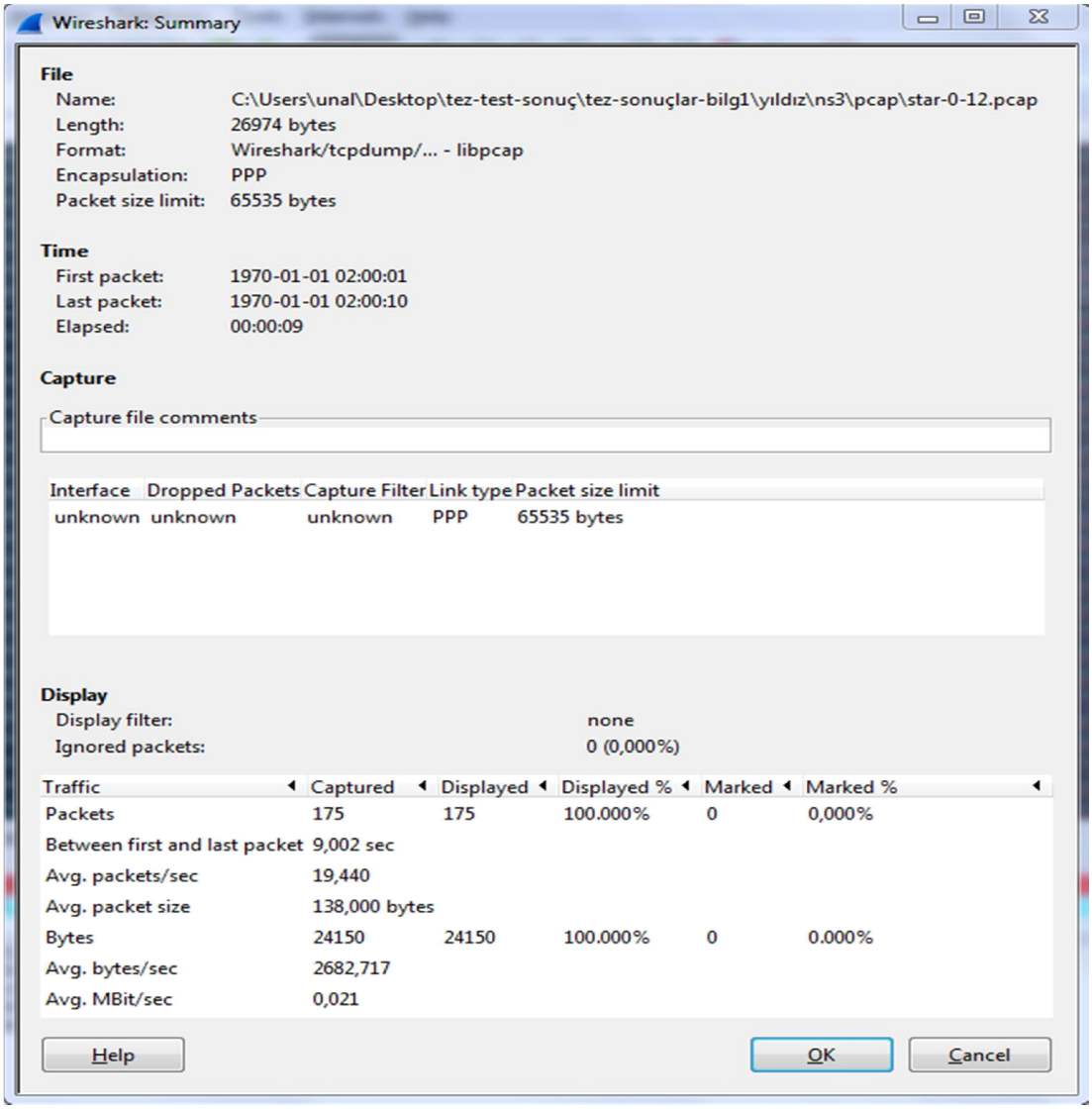
Şekil 6.40. Bilgisayar 1'de NS-3 Yıldız topoloji simülasyon sırasındaki sistem durumu



Şekil 6.41. Bilgisayar 2'de NS-3 Yıldız topoloji simülasyon başlangıcındaki sistem durumu

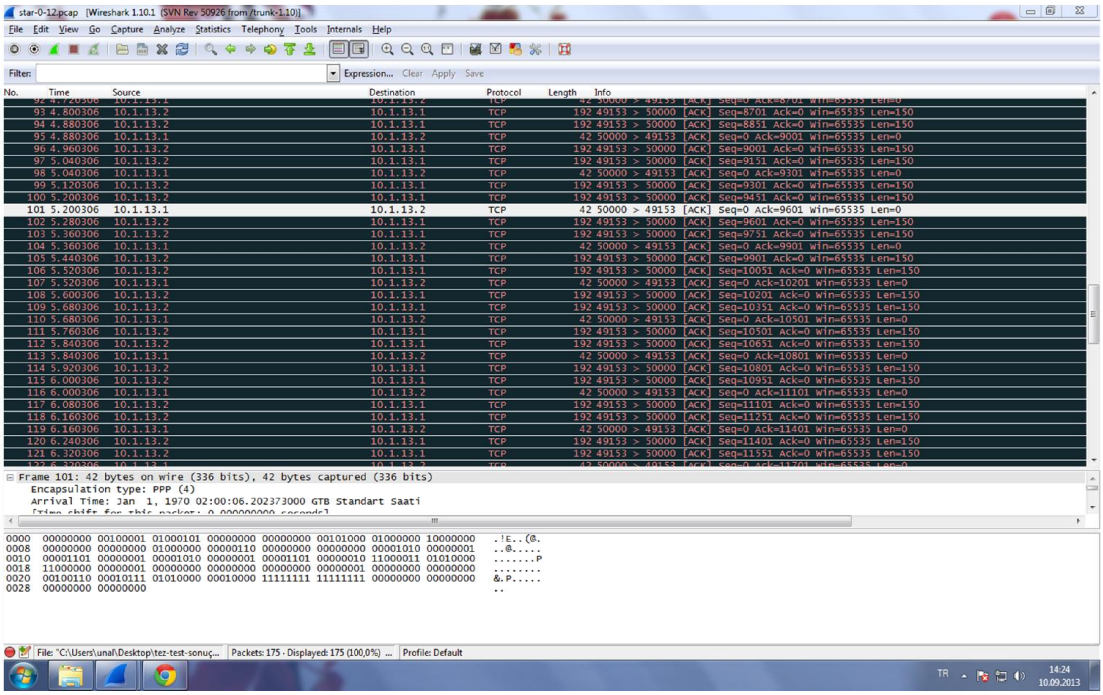
6.2.2.3. NS-3 ağ simülatörü yıldız topoloji ağ çıkış değerleri

NS-3 ağ simülatörü üzerinde, gerçekleştirilen yıldız ağ simülasyonu sonucu, düğümler arasında gerçekleşen trafiği irdelemek için ağ simülatörünün üretmiş olduğu pcap dosyalarını wireshark programını kullanarak açabiliriz. Pcap dosyaları açıldığında simülasyon süresi boyunca gerçekleşen trafik akışına ait olan detaylı paket bilgileri karşımıza çıkacaktır.



Şekil 6.42. NS-3 yıldız topoloji 0-12 düğümleri arasındaki trafik özeti

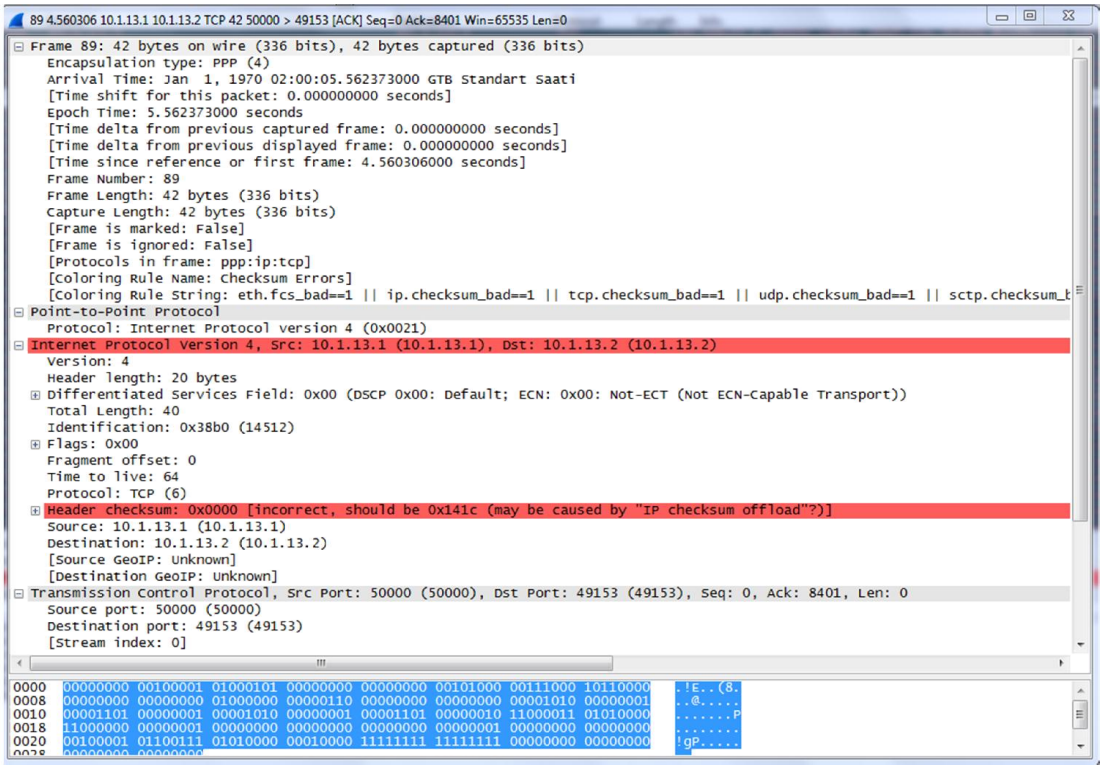
Şekil 6.42'de 0-12 düğümleri arasında gerçekleşen trafiğe ait özet bir bilgi görülmektedir. Simülasyon süresi yakalanan ve gösterilen paket bilgileri ortalama paket boyutu ve ortalama paket adeti bilgileri ve daha bir çok bilgi sunulmaktadır.



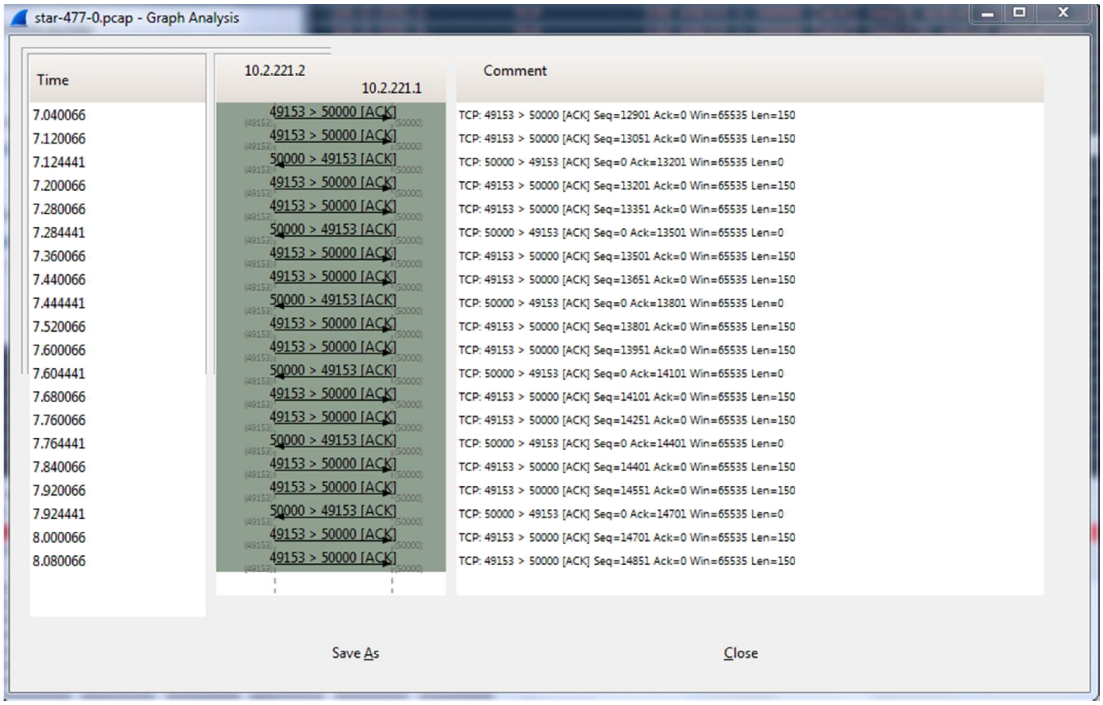
Şekil 6.43. NS-3 yıldız topolojisinde 0-12 düğümleri arası paketler

Şekil 6.43'de 0-12 düğümleri arasında gerçekleşen trafiğe ait yakalanan çerçevelerin listesi gösterilmektedir. Burada incelenmek istenilen çerçeve bilgisi tıklanarak detaylı içeriğe ulaşılabilir. Genel görünümde ise, çerçeve no, çerçevenin yakalanma zamanı, kaynak ve hedef IP adres bilgileri, protokol bilgileri, çerçeve uzunluğu ve bilgi kısmı bulunmaktadır.

Şekil 6.44'de 0-12 düğümleri arasında gerçekleşen trafiğe ait paketler içerisinden 89 nolu çerçeveye ait detaylı içerik görüntülenmektedir. Bu çerçeve içeriğinde çerçeve boyutu IP protokol bilgileri, taşıma katmanı protokolü TCP'ye ait detaylı bilgi ve taşınmakta olan veri ile ilgili bilgiler bulunmaktadır. Bit bazında bu bilgileri görüntülemek ve incelemek mümkündür.



Şekil 6.44. NS-3 yıldız topolojisinde bir pakete ait içerik



Şekil 6.45. NS-3 ızgara topolojisinde 0-12 düğümleri arası akış diagramı

Şekil 6.45'de 0-12 düğümleri arasında gerçekleşen trafiğe ait akış diagramı görülmektedir. Zamana bağlı olarak, hangi düğümler üzerinden akışın gerçekleştiği, IP, port numaraları ve yorum bölümü grafiksel olarak yer almaktadır.

6.2.3. Yıldız ağ modeli için değerlendirme

Tablo 6.6 ve 6.7'de Bilgisayar1 ve Bilgisayar2 üzerinde NS-2 ve NS-3 ağ simülatörleri üzerinde gerçekleştirilen yıldız simülasyonuna ait test sonuçları görülmektedir. Simülasyonların gerçekleştirilmesi ile farklı donanımlar ve simülatörler üzerinde simülasyon süreleri ve simülatörlerin bilgisayar kaynaklarını (işlemci ve bellek) ne kadar kullandığı tespit edilmiştir.

Tablo 6.6. Bilgisayar1 Yıldız topoloji simülasyon sonuçları

SİMULATÖR	SİMULASYON SÜRESİ	KAYNAK KULLANIMI			DÜĞÜM SAYISI
		CPU		BELLEK	
		1	2		
NS-2	27 sn.	%20	%100	%25	500
NS-3	140 sn.	%18	%100	%29	500

Tablo 6.7. Bilgisayar2 Yıldız topoloji Simülasyon Sonuçları

SİMULATÖR	SİMULASYON SÜRESİ	KAYNAK KULLANIMI					DÜĞÜM SAYISI
		CPU				BELLEK	
		1	2	3	4		
NS-2	15 sn.	%14	%09	%08	% 100	%11	500
NS-3	106 sn.	%100	%15	%12	%10	%15	500

Simülatörler için yıldız topoloji örneklerini inceleyecek olursak, ızgara topolojide olduğu gibi, bilgisayar2'nin özellikleri daha iyi olduğu için, NS-2 ve NS-3 uygulaması bilgisayar1'e göre daha kısa zamanda gerçekleşmiştir. Ancak, her iki bilgisayarda da simülasyon NS-3'te daha uzun zamanda gerçekleşmiştir. NS-2 ve NS-3'de simülasyon çalışırken birbirine yakın yüzdelerde işlemci ve bellek kullanımı olmuştur. Her iki bilgisayarda da işlemcinin %100'ü simülasyon çalışırken kullanılmıştır.

6.3. NS-2 &NS-3 Özelliklerini Karşılaştırma

NS-2 ve NS-3 simülatörlerinin karşılaştırılması amacıyla Debajyoti Pal yayımlamış olduğu makalede (Debajyoti, 2012), ağ simülatörlerini paket kaybı, işlem süresi ve bellek kullanımı açısından test etmiştir. Testler sonucu elde edilen sonuçlarda ağ büyüklüğüne bağlı olarak NS-2'de daha fazla paket kaybı gerçekleştiği, NS-3 ün belleği çok daha verimli kullandığı tespit edilmiştir. Ayrıca Debajyoti Pal' ın yapmış olduğu çalışmada NS-2 ve NS-3 simülatörlerinin yanı sıra diğer yaygın olarak kullanılan ağ simülatörlerinin de testleri gerçekleştirilmiş, birbiri ile karşılaştırılarak değerlendirme yapılmıştır. Bu çalışmadaki test sonuçları ile Debajyoti Pal' un yayımlamış olduğu makaledeki sonuçlar kıyaslandığında bellek kullanımı açısından paralel sonuçlar alındığı, işlem zamanı açısından bu çalışmadaki testlerde NS-3 ağ simülatörünün işlemleri NS-2 ağ simülatöründen daha uzun zamanda gerçekleştirdiği tespit edilmiştir.

NS-2 ve NS-3 ağ simülatörlerinin karşılaştırılmasının yapıldığı literatürde yapılmış çalışmalar bulunmaktadır. Juan ve arkadaşları 2010 yılında yapmış oldukları çalışmada, NS-2 ve NS-3 ağ simülatörlerinin mimarisi, dizaynı ve kaynak kodları karşılaştırılmıştır (Juan ve ark., 2010). Zengin ve Çavuşoğlu 2012 yılında yapmış oldukları çalışmada, NS-2 ve NS-3 ağ simülatörlerinin farklı topolojilerde ve donanımlar kullanılarak ölçeklenebilirlik analizi gerçekleştirilmiştir (Zengin ve ark. , 2012).

Tablo 6.8'de NS-2 ve NS-3 ağ simülastörlerine ait bir karşılaştırma tablosu görülmektedir. NS-2 ile ilgili çalışmalar 1996 yılında başlamış ve 2011 yılına kadar değişik sürümler halinde kullanıma sunulmuştur. NS-2 ağ simülatörü NS-1 ağ simülastörünün üzerine eklemeler yapılarak gerçekleştirilmiştir. NS-3 ağ simülatörü ise, 2006 yılında geliştirilmeye başlanmış ve 2008 yılında ilk sürümü yayınlanmıştır. NS-2 ve NS-3 ağ simülatörleri destek bakımından karşılaştırıldığında, NS-3 ağ simülatörüne olan destek her geçen gün artmaktadır.

Tablo 6.8. NS-2 & NS-3 karşılaştırması (Chaudhary ve ark., 2012)

	NS-2	NS-3
İlk sürüm tarihi	1996	2008
Temel yapı	NS-1&REAL	NS-2 GTNets YANS
Mimari	OTCL & C++	C++&PYHTON Scr.
Kurum	DARPA, VINT SAMAN&NSF	NSF CISE & INRIA
Destek	USC ISI & sourceforge, gönüllüler	NSF.INRIA, GT, WashU & , gönüllüler
Script dili	OTCL	Python
Görsel destek	NAM	NS-3-viz, pyviz, NAM, iNSpect
Ölçeklenebilirlik	Sıralı	Dağıtık

6.3.1. Genel yapı

NS-2 ve NS-3 ayrık olay simülatörleridir fakat yapı olarak aralarında ciddi farklılıklar bulunmaktadır. NS-3 gerçek ağ bileşenleri, protokoller ve API'ler ile yapılacak uygulamalara çok daha iyi sonuçlar üretebilmektedir. IP ve UDP protokolleri detaylı olarak analiz edilebilmektedir. NS-3 ün bir diğer avantajı derlenmiş olan kodların tekrar kullanılabilir olmasıdır. NS-3 kod geliştiricileri düşük seviye kod(makine diline yakın) uygulamalar ile soket haberleşmesi, paket iletimi ve adresleme gibi işlemleri çok daha erken safhalarda gerçekleştirebilirler. Tüm bunların sonucu olarak aynı protokol yapıları üzerinde gerçekleştirilen işlemler değerlendirildiğinde iki simülatör arasında yapısal farklar bulunmaktadır. NS-3 gelişmiş yapı ve detaylı analiz özellikleri ile ön plana çıkmaktadır.

6.3.2. Kullanılabilirlik ve adaptasyon

Kullanılabilirlik programlama dili ve derleme açısından değerlendirildiğinde NS-2 C++ ve OTCL üzerinde çalışmaktadır. NS-3 ise sadece C++ dilini kullanır, bu sebepten dolayı üzerinde çalışması ve derlemesi daha kolaydır. Ayrıca NS-3' de isteğe bağlı olarak Python ile de çalışmak mümkündür. NS-2 derleme aşamasında geleneksel GNU derleme sistemini kullanırken NS-3 Python dilinde yazılmış olan

son zamanlarda yaygın olarak kullanılan WAF sistemini kullanır. Dokümantasyon açısından da NS-2'nin yeterli ve tüm modüler yapısını açıklayacak yeterli dökümana sahip olduğu söylenemez fakat NS-3 çok daha genç ve dinamik bir proje olduğundan çok ciddi bir dokümantasyonu ve sürekli olarak artan bir desteğe sahiptir. İki simülatörün modüler yapısına bakıldığında NS-2'nin modülleri arasında kolay bir değişim gerçekleştirmek zordur. NS-3'de katmanlar arası sınırlar ve yapı açık ve net bir şekilde belirlenmiştir. NS-3 uygulama ve ağ katmanında çok daha esnek bir yapıya sahiptir.

6.3.3. Bileşen ve modelleme

NS-3'de yeni bileşen ve düğüm özellikleri ekleme ve oluşturma işlemlerinin çok daha kolay yapılabilmesi amaçlanmıştır. Bu özellik bir ağ simülasyon programı için ciddi ve tercih edilmesini sağlayacak olan bir sebeptir. NS-3 yapısına yeni protokoller düzenli ve kolay bir şekilde eklenebilir, modelleme gerçekleştirilebilir. Özellikle son zamanlarda kablosuz ağ yapıları ile ilgili ciddi çalışmalar bulunmaktadır.

6.3.4. Kurulum, kontrol ve analiz

NS-2 ve NS-3 temel ağ bileşenlerini ağ senaryoları oluşturmak için kullanılmaktadır. Özellikle NS-3 ağ oluşturma, kurulum ve bağlantıların daha kolay yapılabilmesi için çeşitli yardımcı programlar sunarak işlemleri kolaylaştırmaktadır. Ancak her iki simülatörün de veri depolama ve yapılan deneyler üzerinde kontrol imkanları sınırlıdır. NS-3 NS-2'ye göre güçlü bir çerçeveye sahiptir. NS-3 paket izleme için pcap gibi, SQlite gibi harici yazılım izleme ara yüzlerini kullanır. Belli bir standart yapının olması NS-3 için bir avantajdır.

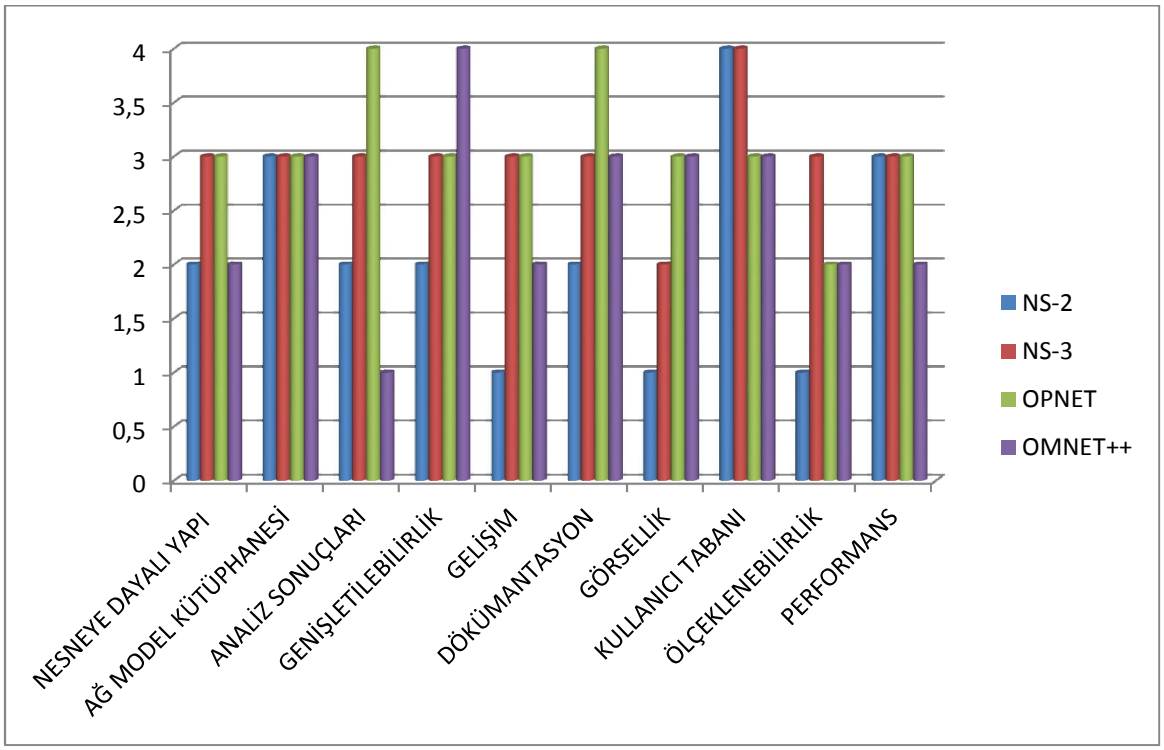
6.3.5. Gelişim durumu

NS-3 projesinin gelişimi boyunca NS-2 sürekli olarak desteklenmiştir fakat NS yazılım geliştirme grubu sadece NS-3 üzerine odaklanmaktadır. NS-2 üzerinde bu gelişim süresince sadece temel güncelleme işlemleri gerçekleştirilmiştir. NS-3 ağ

simülatörünün gelişimi için planlanan NSF projesi ancak 2010 yılı sonu itibariyle tamamlanabilmiştir. NS-3 üzerinde çalışmakta olan bazı modüler yapılar NS-2 üzerinden alınmıştır. Son zamanlarda NS-3 için yeni bir çerçeve geliştirme çalışmasının gerçekleştirileceği NSF tarafından ilan edilmiştir. Bu yapı simülatör üzerinde gerçekleşecek olan simülasyonun sonuçlarının daha iyi bir şekilde analiz edilebilmesi için gerekli geliştirmelerin yapılmasına odaklanacaktır. NS-3 için yapılan geliştirme çalışmasının bir kısmı Google Summer of Code (GSoC) tarafından desteklenmektedir.

6.3.6. Verimlilik ve performans

NS-2 ve NS-3 verimlilik ve performans açısından değerlendirildiğinde, paket kayıpları NS-2 de NS-3'e göre daha fazla olduğu, işlem zamanı ve kaynak kullanımları açısından ise NS-3'ün daha verimli ve performanslı olarak işlemleri gerçekleştirmektedir.



Şekil 6.46. NS-2, NS-3, OPNET, OMNET++ ağ simülatörlerinin karşılaştırılması

Zengin ve arkadaşları gerçekleştirdileri çalışmada, Şekil 6.46 'da görüldüğü üzere 4 adet ağ simülatörünü farklı kriterlere göre değerlendirmesini gerçekleştirmişlerdir. Nesneye dayalı yapı kıyaslamasında, NS-3 ve OPNET ağ simülatörlerinin diğer iki simülatöre göre daha iyi bir yapıya sahip oldukları görülmektedir. Ağ model kütüphanesi açısından bakıldığında ise, bütün ağ simülatörlerinin bir birine yakın bir yapıda oldukları, analiz sonuçlarının değerlendirilmesinde, ticari bir ağ simülatörü olan OPNET ağ simülatörünün diğer açık kaynak simülatörlerinin önüne geçtiği tespit edilmiştir. Genişletilebilirlik özelliği bakımından OMNET++ ağ simülatörünün daha iyi olduğu, ağ simülatörlerinin gelişimi açısından değerlendirildiğinde ise, NS-3 ve OPNET ağ simülatörlerinin daha gelişime açık olduğu görülmüştür. Ağ simülatörlerinin dökümantasyonu açısıdan bakıldığında OPNET ağ simülatörünün en iyi dokümantasyona sahip olduğu, NS-2 ağ simülatörünün ise dökümantasyon noktasında diğer ağ simülatörlerinden zayıf olduğu sonucuna varılmıştır. Görsellik kriterine göre, yine ticari yazılımlar olan OPNET ve OMNET++ ağ simülatörlerinin daha görsel bir yapıya sahip olduğu, açık kaynak kodlu olan ağ simülatörlerinden daha ön plana çıktığı görülmektedir. Ölçeklenebilirlik açısıdan değerlendirildiğinde açık kaynak kodlu NS-3 ağ simülatörünün diğer ağ simülatörlerinden daha iyi bir yapıya sahip olduğu, performans değerlendirmesinde ise, OMNET++ ağ simülatörünün performansının diğerlerinden biraz daha düşük, diğer ağ simülatörünün performansının eşit olduğu tespit edilmiştir.

Sonuç olarak tüm özellikler birlikte değerlendirildiğinde, OPNET ve NS-3 ağ simülatörlerinin diğer simülatörlere göre daha başarılı ve gelişime açık oldukları görülmektedir (Zengin ve ark., 2012).

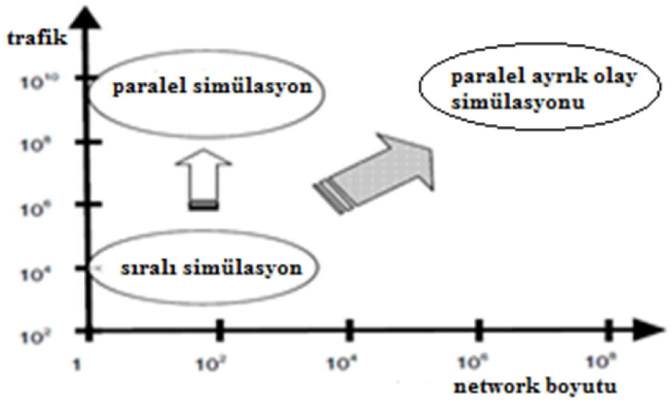
BÖLÜM 7. SONUÇLAR VE DEĞERLENDİRME

Bilgisayar ağlarının simülasyonu için kullanılan birçok simulatör arasından açık kaynak kodlu yazılımlar olan NS-2 ve NS-3 diğer birçok lisanslı ve yüksek maliyetli ürünün yanında büyük ölçekli ağların analiz ve simülasyonunda tercih edilmektedir. NS-3, NS-2'nin tüm özelliklerini bünyesinde barındırmamaktadır. Fakat NS-3 ağ simülatörünü ön plana çıkaran, önemini artıran en önemli nokta özellikle internet tabanlı uygulamalarda yeni ve etkili yeteneklere sahip olmasıdır.

Bu tez çalışmasında NS-2 ve NS-3 ağ simülatörleri üzerinde farklı donanımlar ve ağ topolojileri kullanılarak analizler gerçekleştirilmiştir. Her bir uygulama için farklı donanıma sahip bilgisayarlar üzerinde, simülasyon sırasında ağ simülatörlerine ait işlem süresi, işlemci ve bellek kullanım değerleri, ağ çıkış değerleri gibi değerler ölçülmüştür. Test parametrelerine bağlı olarak uygulanan örneklerde farklı performanslar elde edilmiştir. Simülasyonun gerçekleme süresi değerlendirildiğinde NS-3 simülatörü, NS-2 simülatörüne göre ızgara ağ yapısında ortalama 3-4 kat, yıldız ağ topolojisinde ortalama 6-7 kat daha fazla sürede işlemi tamamlamıştır. İşlem süresi arasındaki bu farkın bir sebebi NS-3'e sonradan eklenen ve NS-2'nin yapısında bulunmayan katmanlara ait işlemlerin yapılmasıdır.

İşlemci kullanımı açısından bakıldığında her iki simülatörün, tüm simülasyon işlemlerinde bir adet işlemcinin nerdeyse tamamına yakınını kullandığı gözlemlenmiştir. Bellek kullanımı açısından değerlendirildiğinde NS-3, NS-2'ye göre tüm simülasyon işlemlerinde belleği daha fazla miktarda kullandığı görülmüştür. NS-3 ağ simülatöründe düğüm yapısı, NS-2 ağ simülatörüne göre daha sadeleştirilmiş ve dinamik bir yapı haline getirilmiş, sadece gerekli olan bileşenler ile ilişkilendirilmiş olduğundan dolayı belleği daha efektif kullanmıştır.

Çalışmada ayrıca ağ simülatörleri ile ilgili yapılmış olan birçok çalışmaya da yer verilmiştir. Bu çalışmalarda elde edilen sonuçlar doğrultusunda NS-2 ve NS-3 ağ simülatörlerinin, diğer ağ simülatörlerine göre karşılaştırılması yapılmıştır. NS-3 ağ simülatörünün eğitimsel amaçlı olarak kullanımda birçok simülatörden üstün özellikler taşıdığı tespit edilmiştir. NS-3 ağ simülatörünün geliştirilmesi için, ciddi bir çalışma söz konusudur. NS-3 ağ simülatörünün eksik modülleri her geçen gün tamamlanarak daha kararlı bir yapıya kavuşmakta, farklı protokollerdeki ağ yapılarının simülasyonuna imkan tanınmaktadır.



Şekil 7.1. Ağ boyutu ve trafik artış grafiği

Şekil 7.1' deki (Fujimoto ve ark., 2003) grafikte sıralı, paralel işlem zamanlı ve paralel ayrık olay simülasyonlarının büyüyen ağ boyutları ile birlikte değerlendirildiğinde ihtiyacın nasıl ortaya çıktığı açıkça görülmektedir. Günümüzün gittikçe büyüyen ve karmaşıklaşan ağ yapılarını simüle edecek ve daha gerçekçi sonuçlar üretip sağlıklı analizlerin ortaya çıkmasını sağlayacak ağ simülatörlerine duyulan ihtiyaç artmaktadır. Ölçeklenebilirlik ve performans açısından değerlendirdiğimiz NS-3 simülatörü birçok açıdan gereksinimleri karşılayacak ve ortaya çıkan yeni ihtiyaç ve talepleri karşılayacak, gelişime açık altyapı ve mimariye sahip olduğu, NS-3 üzerinde geleceğe dönük çalışmaların bu kapsamda devam edeceği sonucuna varılabilir.

KAYNAKLAR

ARNOLD, H. B., KIRK, A. S., Dıscrete event sımulatıon on the world wıde web using java, proceedings of the winter simulation conference, 1996.

ATSAN, E., A Scalable and Reactive Replication Framework For Mobile Ad-hoc Networks, Yüksek lisans tezi, Bilgisayar mühendisliği , Koç üniversitesi, 2007.

BALDO, N., MARCO M., MANUEL R., JAUME N., An open source product-oriented LTE network simulator based on ns-3, In Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems, pp. 293-298. ACM, 2011.

BANKS, J., CARSON, J. S., Discrete-EventSystem Simulation. Englewood Cliffs, New Jersey, Prentice Hall, ISBN 0132155826, 1984.

BECKER, M., TIMM-GIEL, A., MURRAY, K., LYNCH, C., GORG, C., PESCH, D., Comparative Simulations of WSN. In: Cunningham, P., Cunningham, M. (eds.): ICT-MobileSummit 2008. brations. Bulletin of the American Mathematical Society, Vol. 49, No.1, 2008.

CHAUDHARY, R., SETHI, S., KESHARI, R., GOEL, S., A study of comparison of Network Simulator -3 and Network Simulator -2, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3, No.1, pp. 3085–3092, 2012.

CHENG, L., XIN Z., ANU, G. B. , IEEE 802.15. 4 simulation module in network simulator gtnets, Vehicular Technology Conference, 2006, VTC 2006-Spring. IEEE 63rd., Vol. 3, 2006.

CODL, D., KACER, J., KOUTNY, T., Comparison Evaluation of Java Based Discrete Time Simulation Tools, in Proceedings of the 37th International Conference MOSIS 2003: Modeling and Simulation of Systems, pp. 125-130, MARQ, Ostrava, Czech Republic, April 2003.

COURANT, R., Variational methods for the solution of problems of equilibrium, Scalable Simulation Framework API Reference Manual version 1.0, http://www.ssfnet.org/SSFdocs/ssfapiManual.pdf, 1999.

ÇAVUŞOĞLU, Ü., ZENGİN, A., NS-2 ve NS-3 Ağ Simülatörlerinin Ölçeklenebilirlik Analizi ve Karşılaştırma, Bilişim Teknolojileri Dergisi, Vol.5, No. 3, pp. 41-50, 2012.

DANIEL, S. M., Hardware And Software Technıques For Scalable Thousand-Core Systems, Yüksek Lisans Tezi, Stanford Üniversitesi, 2012.

DEVELİ, H., Süleyman Demirel Üniversitesi Kampüs Ağının Opnet ile modellenmesi, Yüksek Lisans Tezi, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü, 2009.

DÖNERTAŞ, Ç., Hareketlilik, Tıkanıklık Ve Çekişmenin IEEE 802.15.4 Tabanlı Ağlardaki Ağın Performansına Olan Etkilerinin İncelenmesi, Yüksek Lisans Tezi, Ege Üniversitesi, Fen Bilimleri Enstitüsü, 2008.

EZREIK, A., ABDALLA G., Design and Simulation of Wireless Network using NS-2, 2nd International Conference on Computer Science and Information Technology (ICCSIT'2012) Singapore, April 28-29 2012.

FAZELI, M., HASAN V., Assessment of Throughput Performance Under OPNET Modeler Simulation Tools in Mobile Ad Hoc Networks (MANETs), Computational Intelligence, Communication Systems and Networks (CICSyN), 2011 Third International Conference on IEEE, 2011.

FENG, T.D., Implementatıon Of BGP In A Network Simulator, Simon Fraser Unıversıty, 2004.

FONT, J.L., PABLO, I., MANUEL, D., JOSÉ, L.S., CLAUDIO, A., Architecture, design and source code comparison of NS-2 and NS-3 network simulators, In Proceedings of the 2010 Spring Simulation Multiconference, Society for Computer Simulation International, 2010.

FUJIMOTO, R., M., Large-scale network simulation: how big? how fast?, Modeling, Analysis and Simulation of Computer Telecommunications Systems, MASCOTS 2003, 11th IEEE/ACM International Symposium on IEEE, 2003.

GUPTA, S. G., Open-Source Network Simulation Tools: An Overview, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Vol.2, No.4, pp-1629, 2013.

HASAN, M.S., HONGNIAN, Y.U., ALISON, G., YANG, T.C., Simulation of Distributed Wireless Networked Control Systems over MANET using OPNET, Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, London, UK, 15-17 Nisan 2007.

HENDERSON, T.R., SUMIT, R., SALLY, F., GEORGE, F.R., NS-3 project goals, In Proceeding from the 2006 workshop on ns-2: the IP network simulator, ACM, 2006.

HUNG-YING, T., Design, realization and evaluation of a component-based compositional software architecture for network simulation**,** Doktora Tezi, The Ohio State University, ISBN:0-493-52811-3, 2002.

JONATHAN, B. H., A Scalable & Extensible Peer-to-Peer Network Simulator, Doktora Tezi, Ottawa-Carleton Institute for Computer Science, Carleton University , Ottawa, Ontario, April 2005.

JONATHAN, B., LAURENT, P., POUL, E.H., Experience Report on Implementing and Simulating a Routing Protocol in NS-2 and NS-3, Advances in System Simulation (SIMUL), 2010 Second International Conference on IEEE, 2010.

JORGE, N., A Comprehensible GloMoSim Tutorial, INRS-Universite du Quebec, The Parallel Computing Laboratory UCLA, 2004.

LOKESH, B, MINEO, T., RAJAT, A., KEN, T., RAJIVE, B., MARIO, G., GloMoSim: A Scalable Network Simulation Environment, Technical Report, Computer Science Department University of California, Los Angeles, Mayıs 1999.

MAEDA, K., KAZUKI, S., KAZUKI, K., AKIKO, Y., AKIRA, U., HIROZUMI, Y., KEIICHI, Y., TERUO H., Getting urban pedestrian flow from simple observation: Realistic mobility generation in wireless network simulation, In Proceedings of the 8th ACM international symposium on Modeling, Analysis and simulation of wireless and mobile systems, pp. 151-158, ACM, 2005.

MATHIEU, L., Outils D'exp_Erimentation Pour La Recherche En R_Eseaux, Doktora Tezi, Unıversıte De Nıce-Sophıa Antıpolıs Scıences Et Technologıes De L'ınformatıon Et De La Communıcatıon, 2010.

MEER, M., The Delta Object Trackıng And Localızatıon Algorıthm For Sensor Networks, Yüksek Lisans Tezi, der Philosophisch-naturwissenschaftlichen Fakult at der Universitat Bern, 2006.

Micheal, J. P., A Comparison Of Sensor Network Simulations, Yüksek Lisans Tezi, Faculty Of Rensselear Polytechnic Enstitüsü, 2004.

MOHAMMAD, A. S., Tasarsız Ağ Yönlendirme Protokollerinin Karşılaştırması, Yüksek Lisans Tezi, İstanbul Üniversitesi, Fen Bilimleri Enstitüsü, Haziran 2006.

MUSLIM, M. Z., FENG, J. G., Gigabit Ethernet Network Design Using OPNET, Lamar University, Yüksek Lisans Tezi, 105 p., Texas, 2001.

NICOL, D., Scalability of network simulators revisited, Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference, 2003.

NICOL, D.M., LIU, J., LILJENSTAM, M., YAN, G., Simulation of large scale networks using SSF. In Simulation Conference, Proceedings of the 2003 Winter ,Vol. 1, pp. 650-657, IEEE, 2003.

NICOL, D.M., MICHAEL L., JASON L., Advanced concepts in large-scale network simulation, Proceedings of the 37th conference on Winter simulation, Winter Simulation Conference, 2005.

PAL, D., A Performance Evaluation of Commonly Used Network Simulators, IJECCE, Vol. 3, No.1, pp. 57-61, 2012.

POLLATSCHEK, M., Programming Discrete Simulations, Publishers Group West, 1996.

RACHNA, C., SHWETA, S., RITA, K., SAKSHI, G., A study of comparison of Network Simulator -3 and Network Simulator -2, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol.3, No.1, pp.3085–3092, 2012.

RAHMAN, M., A., ALGIRDAS, P., FRANK, Z.W., Network modelling and simulation tools, Simulation Modelling Practice and Theory, Vol.17, No.6, pp. 1011-1031, 2009.

RAKESH, S., Design and Implementation of LRC and CRC algorithms in Netsim, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, Vol.2, No.3, pp.1288-1291, Mayıs 2012.

RILEY, G. F., The Georgia Tech Network Simulator, In Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible ,network Research, MoMeTools '03, ACM, New York, 25 - 27 August 2003.

RILEY, G.E., SHARIF, M. L., WENKE, L., Simulating internet worms, Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, (MASCOTS 2004), Proceedings. The IEEE Computer Society's 12th Annual International Symposium on. IEEE, 2004.

RILEY, G.F., AMMAR, M.H., FUJIMOTO, R., Stateless Routing in Network Simulations, In Proceedings of the 8th international Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems MASCOTS. IEEE Computer Society, Washington, DC, 524, August 29 - September 01 2000.

RIMON B., ZYGMUNT, J.H., ROBBERT, V.R., JiST: Embedding Simulation Time into a Virtual Machine, MSWiM ’03 San Diego, California USA, 2003.

RIMON, B., An Efficient Unifying Approach To Simulation Using Virtual Machines, Doktora tezi, Cornell university, 2004.

RIMON, B., ZYGMUNT, J.H., ROBBERT, V.R., Jist: An Efficient Approach To Simulation Using Virtual Machines, Software-Practice And Experience, Vol.35, No.6, pp. 539-576, 2005.

ROBERT, R.H., An Implementation of the SSF Scalable Simulation Framework on the Cray, MTA. PADS, pp.77-88, 2003.

RUBINSTEIN, R.Y., ALEXANDER, S., Discrete event systems: Sensitivity analysis and stochastic optimization by the score function method, New York: Wiley, Vol. 346, 1993.

SARKAR, N.I., SYAFNIDAR, A.H., A review of simulation of telecommunication networks: simulators, classification, comparison, methodologies, and recommendations, Cyber Journals: Multidisciplinary Journals in Science and Technology-Journal of Selected Areas in Telecommunications (JSAT), Vol. 2, No.3, pp. 10-17, 2011.

SIRAJ, S., KUMAR, G.A., BADGUJAR, R., Network Simulation Tools Survey, International Journal of Advanced Research in Computer and Communication Engineering ,Vol. 1, No. 4, Temmuz 2012.

SIRMA, M., Kablosuz Algılayıcı Ağlarının Omnet++ ile Simülasyonu, Yüksek Lisans Tezi, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, 2006.

SOBEIH, A., WEI-PENG, C., HOU, J. C., KUNG, L. C., N. LI, H. YINGTYANI, H. ZHANG, J-Sim: A Simulation Environment for Wireless Sensor Networks, Simulation Symposium, Proceedings. 38th Annual, Sayfa: 175 – 187, 2005.

TAMARA, K., A Simulator for Marine Wireless Sensor Networks, Department of Computing Sciences Texas A&M University, Yüksek lisans tezi, 2011.

URL1, "Simulating Computer Networks with Opnet", http://www.opnet.com /university_program/teaching_with_opnet/textbooks_and_materials/materials/OPNET_Modeler_Tutorial.pdf, Erişim Tarihi: 09.11.2013.

URL2, "Opnet It Guru Online Documentation", http://www.opnet.com/university program/teaching_with_opnet/textbooks_and_materials/materials/ITGAE_Tool_Ntwrk_Ed.pdf, Erişim Tarihi: 07.10.2013.

URL3, "OMNeT++ User Manual Version 4.2.2", http://omnetpp.org/doc/omnetpp/ manual/usman.html, András Varga and OpenSim Ltd., Erişim Tarihi: 14.10.2013.

URL4, "The Autonomous Component Architecture", http://jsim.cs.uiuc.edu/white pap ers/aca.html, Erişim Tarihi: 23.07.2013.

URL5, "The Tcl/Java Project", www.sourceforge.net, Erişim Tarihi: 02.08.2013.

URL6, "J-Sim Project", http://j-sim.cs.uiuc.edu/, Erişim Tarihi: 06.08.2013.

URL7, "Jist-Swans", http://jist.ece.cornell.edu/, Erişim Tarihi: 10.08.2013.

URL8, "Netsim", http://www.tetcos.com/netsimgen.html, Erişim Tarihi: 11.08.2013.

URL9, "Netsim bileşenleri", http://www.tetcos.com/netsim_comp.html, Erişim Tarihi: 12.08.2013.

URL10, "SSFNet", http://www.ssfnet.org/internetPage.html, Erişim Tarihi: 09.08.2013.

URL11, “GTnetS”, http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/index.html, Erişim Tarihi: 15.08.2013.

URL12, “NetAnim”, http://www.nsnam.org/news/netanim-3-103/, Erişim Tarihi: 16.08.2013.

URL13, “NAM(Network Animator)”, http://www.isi.edu/nsnam/nam/, Erişim Tarihi: 14.08.2013.

URL14, “Tracegraph”, http://www.tracegraph.com/download.html, Erişim Tarihi: 18.08.2013.

URL15, “Wireshark”, http://www.wireshark.org/download/docs/user-guide-a4.pdf, Erişim Tarihi: 13.07.2013.

VARGA, A., RUDOLF, H., An overview of the OMNeT++ simulation environment, Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

WANG, A., WENRONG, J., Research of Teaching on Network Course Based on NS-3, Education Technology and Computer Science, ETCS'09First International Workshop on IEEE, Vol. 2, 2009.

WAUPOTITSCH, R., Multi-scale integrated information and telecommunications system (MIITS): first results from a large-scale end-to-end network simülatör, Simulation Conference, WSC 06 Proceedings of the Winter, IEEE, 2006.

WEINGARTNER, E., LEHN, H. V., WEHRLE, K., A performance comparison of recent network simulators, In Proc. ICC, pp.1-5, 2009.

WITOLD, D., STEFFEN, S., VLADO, H., ANDREAS, K., A Mobility Framework for OMNeT++, Telecommunication Networks Group, Technische Universität Berlin, Ocak 2003.

XIAO-JIAN, L., CHUN-HE, X., Lİ, L., HAI-QUAN, W., A scenario description language of network attack and defense, In Proceedings of the IASTED Asian Conference on Modelling and Simulation, ACTA Press, pp. 150-156, Ekim 2007.

XINJIE, C., Network Simulations With Opnet, Proceedings of the 1999 Winter Simulation Conference, 1999.

YOON, S., YOUNG, B., K., A design of network simulation environment using ssfnet." Advances in System Simulation, SIMUL'09. First International Conference on. IEEE, 2009.

YUN, J., KIWOOK, S., HYUNSOO, Y., Dynamic Simulation on Network Security Simulator Using SSFNET, Convergence Information Technology, International Conference on. IEEE, 2007.

ZEIGLER, B. P., PRAEHOFER, H., KIM, T. G., Theory of Modeling and Simulation, New York: Academic Press., 2000.

ZEIGLER, B.P., SAURABH, M., Modeling and Simulation of Ultra Large Networks: A Framework for New Research Directions, supported by NSF Grant ANI-0135530, ULN Workshop, 2002.

ZENG, X., BAGRODIA, R., GERLA, M., GloMoSim: a library for parallel simulation of large-scale wireless networks, Parallel and Distributed Simulation, PADS 98 Proceedings. Twelfth Workshop, pp.154-161, DOI:10.1109 /PADS. 1998.685281, Mayıs 1998.

ZENGİN, A., ÇAVUŞOĞU, Ü., SEVİN, A., KAÇAR, S., Sistem ve Ağ Mühendisliği Eğitiminde Ağ Simülasyon Teknolojileri, Geleceğin Mühendislik Eğitiminde Endüstri İle İşbirliği Sempozyumu 2012, Isparta, 1-2 Kasım 2012.

ZENGİN, A., EKİZ, H., ÇOBANOĞLU, B., TUNCEL, S., A Simulation Study Of The OSPF Protocol In Devs- Suite, ETAI, 26-29 September, Ohrid, Macedonia, 2009.

ZENGİN, A., Modeling discrete event scalable network systems, Information Sciences 181, No. 5, pp.1028-1043, 2011.

ZHAO, W., JIANG X., OPNET-based modeling and simulation study on handoffs in Internet-based infrastructure wireless mesh networks, Computer Networks, Vol. 55, No. 12, pp. 2675-2688, 2011.

EKLER

EK-1 NS-2 Kurulumu

İnternet bağlantısı olan bilgisayarda terminale aşağıdaki komutlar sırasıyla yazılarak NS-2 indirilip, kurma işlemi yapılabilir:

wget http://nchc.dl.sourceforge.net/sourceforge/nsnam/ns-allinone-2.34.tar.gz

Bu komut NS-2 yi internetten indirir. Daha sonra kurulum için aşağıdaki komutlar sırası ile terminale yazılır.

tar -xzvf ns-allinone-2.34.tar.gz

cd ns-allinone-2.34

sudo apt-get install build-essential autoconf automake libxmu-dev gcc-4.3

Daha sonra ns-allinone-2.34/otcl-1.13/Makefile.in yolunu izlenerek Makefile.in dosyası içerisindeki CC= @CC@ satırı aşağıdaki satırla değiştirilir.

CC= gcc-4.3

Terminale dönüp "./install" yazılır. Bu işlem bittiğinde

gedit ~/.bashrc

komutu ile açılan dosyaya aşağıdaki satırlar yapıştırılır, burada dikkat edilecek nokta

LD_LIBRARY_PATH

OTCL_LIB=/your/path/ns-allinone-2.34/otcl-1.13

NS2_LIB=/your/path/ns-allinone-2.34/lib

X11_LIB=/usr/X11R6/lib

USR_LOCAL_LIB=/usr/local/lib

```
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LI
B:$USR_LOCAL_LIB

# TCL_LIBRARY
TCL_LIB=/your/path/ns-allinone-2.34/tcl8.4.18/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# PATH
XGRAPH=/your/path/ns-allinone-2.34/bin:/your/path/ns-allinone-
2.34/tcl8.4.18/unix:/your/path/ns-allinone-2.34/tk8.4.18/unix
NS=/your/path/ns-allinone-2.34/ns-2.34/
NAM=/your/path/ns-allinone-2.34/nam-1.14/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

your/path yazan yeri home/sizin bilgisayarınızın adı ile değiştirmek bu işlem için Ctrl+h ı kullanabilirsiniz. Sonra bu dosya kaydedilir. Terminale dönüp source ~/.bashrc
yazılır ve kurulum işlemi tamamlanır. Terminale ns yazdığınızda % işareti çıkıyorsa program doğru çalışıyor demektir.

EK-2 NS-3 Kurulumu

EK-2 bölümünde, NS-3 ağ simülatör programının kurulumu ile ilgili adımlar sıralanarak detaylandırılmıştır. Öncelikle kurulum için gerekli ön hazırlıkların yapılması, kurulum dosyalarının mercurial kullanılarak indirilmesi ve yüklenmesi adımları sırasıyla detaylandırılmıştır. Daha sonra WAF kullanılarak konfigürasyonların yapılması kodların derlenmesi ve son olarak da kurulumun doğrulanması için özel hazırlanmış Phyton kodundan faydalanarak kurulum testinin yapılması anlatılmıştır. Tüm adımlar tamamlandığında NS-3 simülatörünün kullanımı ve simülasyon için kod geliştirmeye başlayabilirsiniz.

NS-3 İçin Gerekli Paketlerin Kurulumu

Bu tez çalışmasında Ubuntu 12.04 versiyonu kullanılmıştır. Aşağıda NS-3 için gerekli paketler ve bu paketlerin kurulumu için gerekli kod satırları listelenmiştir.

C++ için minimum gereksinim: Tarball kullanılarak ns-3'ün çalıştırılabilmesi için gereken paket kümesini içermektedir.

sudo apt-get install gcc g++ python

Phyton için minimum gereksinim: yayınlanmış bir Tarball'ı phyton kullanarak çalıştırmak için gereken paket kümesini içermektedir.

sudo apt-get install gcc g++ python python-dev

NS-3 kaynak kodu yönetimi için gerekli olan Mercurial yüklenmesi gerekmektedir.

sudo apt-get install mercurial

Phtyon bağlantılarını çalıştırmak için Ns-3 geliştirme ağacı bazaar'a ihtiyaç duymaktadır.

sudo apt-get install bzr

Hata Ayıklama (Debugging)

sudo apt-get install gdb valgrind

GNU Kütüphanesi: Wifi hata modellerini desteklemek için kurulmalıdır.

sudo apt-get install gsl-bin libgsl0-dev libgsl0ldbl

The Network Simulation Cradle (nsc); the flex lexical analyzer ve bison parser generator'e ihtiyaç duymaktadır.

sudo apt-get install flex bison

Bazı Network Simulation Cradle yığınları için gcc-3.4 kurulması gerekmektedir.

sudo apt-get install g++-3.4 gcc-3.4

Pcap paket izlerini okuyabilmek için.

sudo apt-get install tcpdump

İstatistiksel framework'e veritabanı desteği için

sudo apt-get install sqlite sqlite3 libsqlite3-dev

XML tabanlı konfigürasyon deposu için libxml2 >= version 2.7 gerekmektedir.

sudo apt-get install libxml2 libxml2-dev

GTK tabanlı konfigürasyon sistemi

sudo apt-get install libgtk2.0-0 libgtk2.0-dev

Sanal makine ve ns-3 ile deneme yapabilmek için

sudo apt-get install vtun lxc

utils/check-style.py Kod stillerini desteklemek için

sudo apt-get install uncrustify

Satır içi belgelendirme yapabilmek için Doxygen

sudo apt-get install doxygen graphviz imagemagick

sudo apt-get install texlive texlive-pdf texlive-latex-extra texlive-generic-extra texlive-generic-recommended

NS-3 Kullanım klavuzu

sudo apt-get install python-sphinx dia texlive texlive-pdf texlive-latex-extra texlive-extra-utils texlive-generic-recommended

pyviz visualizer

sudo apt-get install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev

openflow modülünü desteklemek için

sudo apt-get install libboost-signals-dev libboost-filesystem-dev

Mercurial Kullanarak NS-3 İndirilmesi

Öncelikle ana dizin altında **repos** isimli bir klasör oluşturunuz. Bu dizine yerel makinenize Mercurial depolarından veri senkronizasyonu için kullanılacaktır. Ns-3-allinone paketinin bir kopyasını oluşturduğunuz dizine kopyalamak için Ubuntu'da bulunan uç birim (terminal) çalıştırarak aşağıdaki komut satırını yazınız.

```
cd
mkdir repos
cd repos
hg clone http://code.nsnam.org/ns-3-allinone
```

hg (Mercurial) komutu çalıştığında aşağıdakine benzer bir çıktı göreceksiniz.

```
destination directory: ns-3-allinone
requesting all changes
adding changesets
adding manifests
adding file changes
added 26 changesets with 40 changes to 7 files
7 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

Kopya alma işlemi tamamlandıktan sonra ~/repos dizini altında ns-3-allinone isminde bir klasör oluştuğunu göreceksiniz. Bu dizinin içeriği aşağıdaki gibi olacaktır.

build.py constants.py dist.py* download.py* README util.py*

Burada birkaç Phyton scripti indirildiğini gözlemleyeceksiniz. Bir sonraki adımda bu scriptleri kullanarak NS-3'ün istediğiniz versiyonunu indirmek ve kurmak için kullanılacaktır. Eğer http://code.nsnam.org web sitesine giderseniz, burada birçok NS-3 versiyonunu bulabilirsiniz. En son yayımlanan NS-3 dağıtımını görmek için http://code.nsnam.org/ns-3-dev/ web sitesini ziyaret edebilirsiniz.
En yaygın seçeneklerle son güncel sürümü indirmek için terminalinize aşağıdaki komut satırını giriniz.

./download.py -n ns-3-dev

Yükleme işlemi tamamlandıktan sonra, ~/repos/ns-3-allinone dizini altında aşağıdaki gibi bir çok dosya ve dizinin oluştuğunu göreceksiniz. ~/repos/ns-3-allinone dizini içerisindeki ns-3-dev klasörüne göz atın. Aşağıdaki gibi dosyaları göreceksiniz.

AUTHORS examples/ RELEASE_NOTES utils/ wscript
bindings/ LICENSE samples/ VERSION wutils.py
*CHANGES.html ns3/ scratch/ waf**
*doc/ README src/ waf.bat**

NS-3 dağıtımını kurmak için gerekli paketler hazır hale gelmiştir.

NS-3'ün NS-3-Allinone Kullanılarak Yüklenmesi

NS-3 kurulumunu ilk kez yapacaklar için hazırlanan ns-3-allinone paketi sayesinde genel ayarların hazır halde olduğu paketi kullanabilirsiniz. Dosyaları indirdiğiniz dosya içerisine geri dönün. Yani Mercurial ile indirmiş olduğunuz ~/repos/ns-3-allinone dizinine geliniz. Terminalinize aşağıdaki komut satırını yazınız.

./build.py

Birçok derleme işlemi gerçeleşecektir ve ardından aşağıdaki gibi son satırları görmelisiniz.

Build finished successfully (00:02:37)
Leaving directory `./ns-3-dev'

WAF Kullanarak Konfigürasyon Yapılandırılması

NS-3 kaynak kodunu yerel bilgisayarınıza indirdikten sonra kullanılabilir programlar üretmek için kaynak kodununun derlenmesi gerekmektedir. NS-3 projesi kod derleme sistemi olarak WAF kullanmaktadır. Waf, Phyton tabanlı yeni nesil derleyicilerden birisidir. NS-3 kodunu derlemek için herhangi bir Phyton dilini

anlamak zorunda değilsiniz, bu işlemi Waf sizin için yapacaktır. Waf hakkında daha detaylı bilgi almak için http://code.google.com/p/waf/ adresinin ziyaret edebilirsiniz.

Geçerli yapılandırma seçeneklerini görmek için terminalinize ./waf --help yazınız. Burada en önemli seçenek –d <debug level> dir. Geçerli hata ayıklama seçenekleri "debug" yada "optimized" dir. Bu değerleri değiştirmek aşağıdaki gibi mümkündür.

CXXFLAGS="-O3" ./waf configure

Ya da alternatif olarak gcc derleyicisi kullanarak aşağıdaki şekilde de yapılabilmektedir.
CXX=g++-3.4 ./waf configure

Not: Bazı diğer yükleme araçları ile geçerli hata ayıklama seçeneğini değiştirmek için aşağıdaki gibi komut satırı kullanmak gerekebilir.
./waf -d optimized configure; ./waf

Derleme sonucunda oluşacak olan binary dosyalar build/<debuglevel>/srcpath diznine yerleşirler. Örneğin, first.cc isminde bir script dosyanızı derlediğinizi varsayalım. Bu kod dosyanızının çalıştırılabilir halini build/debug/first dizninde bulabilirsiniz. Yada çalıştırılabilir dosyayı aşağıdaki komut satırlarıyla hata ayıklama yapabilirsiniz.

```
./waf --shell
 cd build/debug/examples
 gdb first
```

Hata ayıklama için elbette gdb kullanabilirsiniz yada daha popüler hata ayıklayıcı olan ddd veya insight kullanabilirsiniz. Eğer Phyton bağlarını devre dışı bırakılmasını istiyorsanız aşağıdaki komut satırını kullanmalısınız.
./waf --disable-python configure

Gerekirse sudo programı kullanılarak sistemin yapılandırılması istenirse, aşağıdaki komut satırı ile bu opsiyon sağlanabilmektedir.
./waf --enable-sudo configure

Yapılandırmayı en başına getirmek için ise aşağıdaki komut satırı kullanılmaktadır.

./waf distclean

Yada herşeyin başarısız olduğu durumlarda
rm -rf build

Tüm yapılandırmayı eski haline döndürmek için aşağıdaki komut satırı kullanılır.
./build.py

Böylelikle yapılandırma durumunuz sıfırlanacaktır.
Tüm waf seçeneklerini görmek için aşağıdaki komut satırını kullanınız.
./waf --help

NS-3 kurulumunuzun hatasız bir şekilde başarı ile kurulduğunu görebilmek için hazırlanmış olan Phyton scriptini terminalinizden çalıştırmanız gerekmektedir. Bunun için aşağıdaki kod satırını yazmalısınız.
./test.py

Komut sonucunda karşınıza aşağıdakine benzer bir çıktı görüntülenecektir.

PASS: TestSuite histogram
PASS: TestSuite ns3-wifi-interference
PASS: TestSuite ns3-tcp-cwnd
PASS: TestSuite ns3-tcp-interoperability
PASS: TestSuite sample
...

EK-3 Izgara topoloji NS-2 Kodu

NS-2 Ağ simülastörü üzerinde ızgara topoloji simülasyonunu gerçekleştirmek üzere hazırlanan kodlar aşağıda verilmiştir.

ızgara.tcl dosyası

```
#Simulatör nesnesinin oluşturulması
set ns [new Simulator]
set s 16

#NAM iz dosyasının açılması
set nf [open out.nam w]
$ns namtrace-all $nf

set izleme [open ızgara.tr w]
$ns trace-all $izleme

set Uret [new RNG]
$Uret seed 1

#bitirme prosedürünün tanımlanması
proc finish {} {
        global ns nf izleme
        $ns flush-trace
        #NAM iz dosyasının kapatılması
        close $nf
         close $izleme
        #NAM çıktısının oluşturulması
        exec nam out.nam &
        exit 0
}

#düğümlerin oluşturulması
for {set i 0} {$i<[expr $s*$s+1]} {incr i} {
set n($i) [$ns node]
}

#düğümler arasında bağlantı oluşturulması
for {set j 0} {$j<$s} {incr j} {
for {set i 1} {$i<$s} {incr i} {
$ns duplex-link $n([expr $j*$s +$i]) $n([expr ($j*$s +$i+1)]) 5Mb
2ms DropTail
$ns duplex-link-op $n([expr $j*$s +$i]) $n([expr ($j*$s +$i+1)])
orient right
$ns duplex-link-op $n([expr $j*$s +$i]) $n([expr ($j*$s +$i+1)])
color "blue"
} }
for {set k 0} {$k< $s} {incr k} {
```

```
for {set l 0} {$l< $s-1} {incr l} {
$ns duplex-link $n([expr $s*$l+$k+1]) $n([expr ($l+1)*$s+1+$k]) 5Mb
2ms DropTail
$ns duplex-link-op $n([expr $s*$l+$k+1]) $n([expr ($l+1)*$s+1+$k])
orient down
$ns duplex-link-op $n([expr $s*$l+$k+1]) $n([expr ($l+1)*$s+1+$k])
color "red"
} }
for {set x 0} {$x<100} {incr x} {
#kaynak
set kaynak [new RandomVariable/Uniform]
$kaynak use-rng $Uret
$kaynak set min_ 1
$kaynak set max_ [expr ($s*$s)]

#hedef
set hedef [new RandomVariable/Uniform]
$hedef use-rng $Uret
$hedef set min_ 1
$hedef set max_ [expr ($s*$s)]
set t [expr int([$kaynak value])]
set z [expr int([$hedef value])]

#UDP bağlantısı kurulması
set udp($x) [new Agent/UDP]
set null($x) [new Agent/Null]
$ns attach-agent $n($t) $udp($x)
$ns attach-agent $n($z) $null($x)
}
for {set x 0} {$x<100} {incr x} {
$ns connect $udp($x) $null($x)
}
#UDP bağlantısı üzerinde CBR kurulması
for {set x 0} {$x<100} {incr x} {
set cbr($x) [new Application/Traffic/CBR]
$cbr($x)  attach-agent $udp($x)
$cbr($x) set type_ CBR
$cbr($x) set packet_size_ 512
$cbr($x) set rate_ 0.5mb
}

for {set i 1} {$i<100} {incr i} {

#CBR için olay programlanması
$ns at 0.1 "$cbr($i) start"
$ns at 10.0 "$cbr($i) stop"
}
#simülasyonun çalıştırılması
$ns run
```

EK-4 Izgara topoloji NS-3 Kodu

NS-3 Ağ simülastörü üzerinde ızgara topoloji simülasyonunu gerçekleştirmek üzere hazırlanan kodlar aşağıda verilmiştir.

grid-tpl.cc dosyası

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
#include "ns3/csma-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/netanim-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-layout-module.h"

using namespace ns3;

int main (int argc, char *argv[])
{
  Config::SetDefault ("ns3::OnOffApplication::PacketSize",
UintegerValue (512));
  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue
("500kb/s"));

  uint32_t xSize = 16;
  uint32_t ySize = 16;
  std::string animFile = "gd-animation.xml";

  CommandLine cmd;
  cmd.AddValue ("xSize", "Number of rows of nodes", xSize);
  cmd.AddValue ("ySize", "Number of columns of nodes", ySize);
  cmd.AddValue ("animFile",  "File Name for Animation Output",
animFile);

  cmd.Parse (argc,argv);
  if (xSize < 1 || ySize < 1 || (xSize < 2 && ySize < 2))
    {
      NS_FATAL_ERROR ("Need more nodes for grid.");
    }

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue
("5Mbps"));
```

```
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));


  // gridin oluşturulması
  PointToPointGridHelper grid (xSize, ySize, pointToPoint);

  // stack kurulumu
  InternetStackHelper stack;
  grid.InstallStack (stack);

  // IP adreslerinin atanması
  grid.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0",
"255.255.255.0"),
                            Ipv4AddressHelper ("10.2.1.0",
"255.255.255.0"));

 OnOffHelper clientHelper ("ns3::UdpSocketFactory", Address ());
  clientHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  clientHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
  ApplicationContainer clientApps;

    {
      for (int t= 1 ; t < 16;  t+=2 )
         {
           for ( int k = 1 ; k < 16; k+=3)

            {
  AddressValue remoteAddress (InetSocketAddress (grid.GetIpv4Address
(xSize-t ,ySize-k), 1000));
  clientHelper.SetAttribute ("Remote", remoteAddress);

  clientApps.Add (clientHelper.Install (grid.GetNode (t-1,k-1)));

               } }

  clientApps.Start (Seconds (0.0));
  clientApps.Stop (Seconds (10));

  // animasyon için sınırların belirlenmesi
  grid.BoundingBox (1, 1, 100, 100);

  // animasyon objesinin oluşturulması
  AnimationInterface anim (animFile);

  // simülasyonun kurulması
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

 PointToPointHelper p2p;
  CsmaHelper csma;

  p2p.EnablePcapAll ("grid-animation");
  csma.EnablePcapAll ("grid-animation", false);

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
} }
```

EK-5 Yıldız Topoloji NS-2 Kodu

NS-2 Ağ simülastörü üzerinde yıldız topoloji simülasyonunu gerçekleştirmek üzere hazırlanan kodlar aşağıda verilmiştir.

yıldız.tcl dosyası

```
#simülatör nesnesinin oluşturulması
set ns [new Simulator]

#iz dosyasını açılması
set izleme [open starizleme.tr w]

#nam'a ait dosyanın açılması
set starnam [open starnam.nam w]
$ns trace-all $izleme
$ns namtrace-all $starnam
set uretec [new RNG]
$uretec seed 0

#topolojinin kurulumu
for {set i 0} {$i<500} {incr i} {
set d($i) [$ns node] }
for {set j 0} {$j<498} {incr j} {
$ns duplex-link $d($j) $d(498) 5Mb 2ms DropTail }
$ns duplex-link $d(498) $d(499) 5Mb 2ms DropTail

#Trafik oluşturulması
for {set k 0} {$k<498} {incr k} {
set tcp($k) [new Agent/TCP]
set ftp($k) [new Application/FTP]

$ns attach-agent $d($k) $tcp($k)
set tcp_alici($k) [new Agent/TCPSink]
$ns attach-agent $d(499) $tcp_alici($k)
$tcp($k) set packet_size 150
$tcp($k) set rate 0.015mb
$ns connect $tcp($k) $tcp_alici($k)
$ftp($k) attach-agent $tcp($k)
}

#iletimin başlatılması
for {set l 0} {$l<498} {incr l} {
$ns at 0.0 "$ftp($l) start"
$ns at 10.0 "$ftp($l) stop"
}
```

```
# sonlandırma prosedürü
proc finish {} {global ns izleme starnam
               $ns flush-trace
            #izleme dosyasının kapatılması
               close $izleme
            #nam dosyasının kapatılması
               close $starnam
            # üretilen nam dosyasının çalıştırılması
               exec nam starnam.nam &
             exit 0

}
$ns at 11 "finish"
$ns run
```

EK-6 Yıldız Topoloji NS-3 Kodu

NS-3 Ağ simülatörü üzerinde yıldız topoloji simülasyonunu gerçekleştirmek üzere hazırlanan kodlar aşağıda verilmiştir.

yldz-tpl.cc dosyası

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
#include "ns3/csma-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/netanim-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-layout-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("StarAnimation");

int
main (int argc, char *argv[])
{

  Config::SetDefault ("ns3::OnOffApplication::PacketSize",
UintegerValue (150));

  // data rate-- 15kb/s
  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue
("15kb/s"));

  // node sayısı

  uint32_t nSpokes = 500;
  std::string animFile = "star-animation.xml";
  uint8_t useIpv6 = 0;
  Ipv6Address ipv6AddressBase = Ipv6Address("2001::");
  Ipv6Prefix ipv6AddressPrefix = Ipv6Prefix(64);

  CommandLine cmd;
  cmd.AddValue ("nSpokes", "Number of spoke nodes to place in the
star", nSpokes);
  cmd.AddValue ("animFile",  "File Name for Animation Output",
animFile);
  cmd.AddValue ("useIpv6",   "use Ipv6", useIpv6);
```

```
  cmd.Parse (argc, argv);

  NS_LOG_INFO ("Build star topology.");
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue
("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  PointToPointStarHelper star (nSpokes, pointToPoint);

  NS_LOG_INFO ("Install internet stack on all nodes.");
  InternetStackHelper internet;
  star.InstallStack (internet);

  NS_LOG_INFO ("Assign IP Addresses.");
  if (useIpv6 == 0)
    {
      star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0",
"255.255.255.0"));
    }
  else
    {
      star.AssignIpv6Addresses (ipv6AddressBase, ipv6AddressPrefix);
    }

  NS_LOG_INFO ("Create applications.");
  //
  // paketlerin oluşturulması
  //
  uint16_t port = 50000;
  Address hubLocalAddress;
  if (useIpv6 == 0)
    {
      hubLocalAddress = InetSocketAddress (Ipv4Address::GetAny (),
port);
    }
  else
    {
      hubLocalAddress = Inet6SocketAddress (Ipv6Address::GetAny (),
port);
    }
  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
hubLocalAddress);
  ApplicationContainer hubApp = packetSinkHelper.Install
(star.GetHub ());
  hubApp.Start (Seconds (0.0));
  hubApp.Stop (Seconds (10.0));

  // uygulamanın oluşturulması

  OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
  onOffHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));

  ApplicationContainer spokeApps;

  for (uint32_t i = 0; i < star.SpokeCount (); ++i)
    {
      AddressValue remoteAddress;
```

```
      if (useIpv6 == 0)
        {
          remoteAddress = AddressValue(InetSocketAddress
(star.GetHubIpv4Address (i), port));
        }
      else
        {
          remoteAddress = AddressValue(Inet6SocketAddress
(star.GetHubIpv6Address (i), port));
        }
      onOffHelper.SetAttribute ("Remote", remoteAddress);
      spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
    }
  spokeApps.Start (Seconds (1.0));
  spokeApps.Stop (Seconds (10.0));

  NS_LOG_INFO ("Enable static global routing.");


  if (useIpv6 == 0)
    {
      Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
    }

  // animasyon için sınırların belirlenmesi
  star.BoundingBox (1, 1, 100, 100);

  // animasyon nesnesinin oluşturulması
  AnimationInterface anim (animFile);


  PointToPointHelper p2p;
  CsmaHelper csma;

  AsciiTraceHelper ascii;
  Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream
("star.tr");
  p2p.EnableAsciiAll (stream);
  csma.EnableAsciiAll (stream);

  p2p.EnablePcapAll ("star");
  csma.EnablePcapAll ("star", false);


  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");

  return 0;
}
```

ÖZGEÇMİŞ

Ünal ÇAVUŞOĞLU, 18.11.1981 tarihinde Akhisar'da doğdu. İlkokul – ortaokul ve lise eğitimini Akhisar'da muhtelif okullarda tamamladı. 2007 yılında başladığı Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2011 yılında mezun oldu. 2011 yılında Sakarya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar ve Bilişim Mühendisliği Anabilim Dalı'nda yüksek lisansa başladı. Sakarya Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü'nde 2011 yılında araştırmacı olarak göreve başladı. Bilgisayar ağları, veri madenciliği, yazılım mühendisliği, veri tabanı yönetim sistemleri konusunda çalışmaları bulunmaktadır.