MISSOURI S&T

CENTER FOR INFRASTRUCTURE ENGINEERING STUDIES

BTMA Protocol Implementation in NS2

by

Yi Wang

UTC
R165

A University Transportation Center Program
at Missouri University of Science & Technology

Disclaimer

The contents of this report reflect the views of the author(s), who are responsible for the facts and the accuracy of information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program and the Center for Infrastructure Engineering Studies UTC program at the Missouri University of Science & Technology, in the interest of information exchange. The U.S. Government and Center for Infrastructure Engineering Studies assumes no liability for the contents or use thereof.

Technical Report Documentation Page

1. Report No. UTC R165	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle BTMA Protocol Implementation in NS2		5. Report Date September 2008 6. Performing Organization Code	
7. Author/s Yi Wang		8. Performing Organization Report No. 00010050	
9. Performing Organization Name and Address Center for Infrastructure Engineering Studies/UTC program Missouri University of Science & Technology 220 Engineering Research Lab Rolla, MO 65409		10. Work Unit No. (TRAIS) 11. Contract or Grant No. DTRS98-G-0021	
12. Sponsoring Organization Name and Address U.S. Department of Transportation Research and Special Programs Administration 400 7th Street, SW Washington, DC 20590-0001		13. Type of Report and Period Covered Final 14. Sponsoring Agency Code	
15. Supplementary Notes			
16. Abstract BTMA protocol is a new protocol designed to mitigate hidden terminal problem in wireless communication. However, the challenging problem of implementing it in NS2 is there is only one physical channel in original NS2 mobile models. In order to add more wireless channels, I designed a new approach to simulate in multiple channels between mobile nodes in NS2.			
17. Key Words Design, Education, Future Transportation Professionals	18. Distribution Statement No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.		
19. Security Classification (of this report) unclassified	20. Security Classification (of this page) unclassified	21. No. Of Pages 5	22. Price

Form DOT F 1700.7 (8-72)

BTMA Protocol Implementation in NS2

1. Background

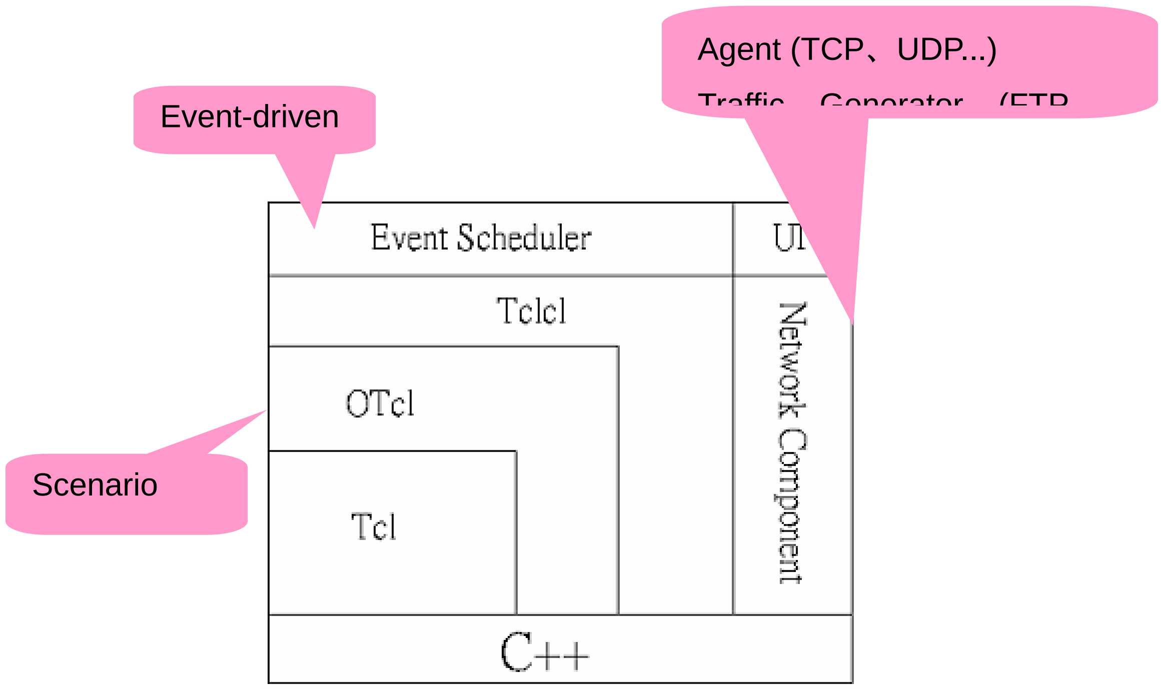
BTMA protocol is a new protocol designed to mitigate hidden terminal problem in wireless communication. However, the challenging problem of implementing it in NS2 is there is only one physical channel in original NS2 mobile models. In order to add more wireless channels, I designed a new approach to simulate in multiple channels between mobile nodes in NS2.

(1).NS2 Architecture.

NS2 meets both of these needs with two languages

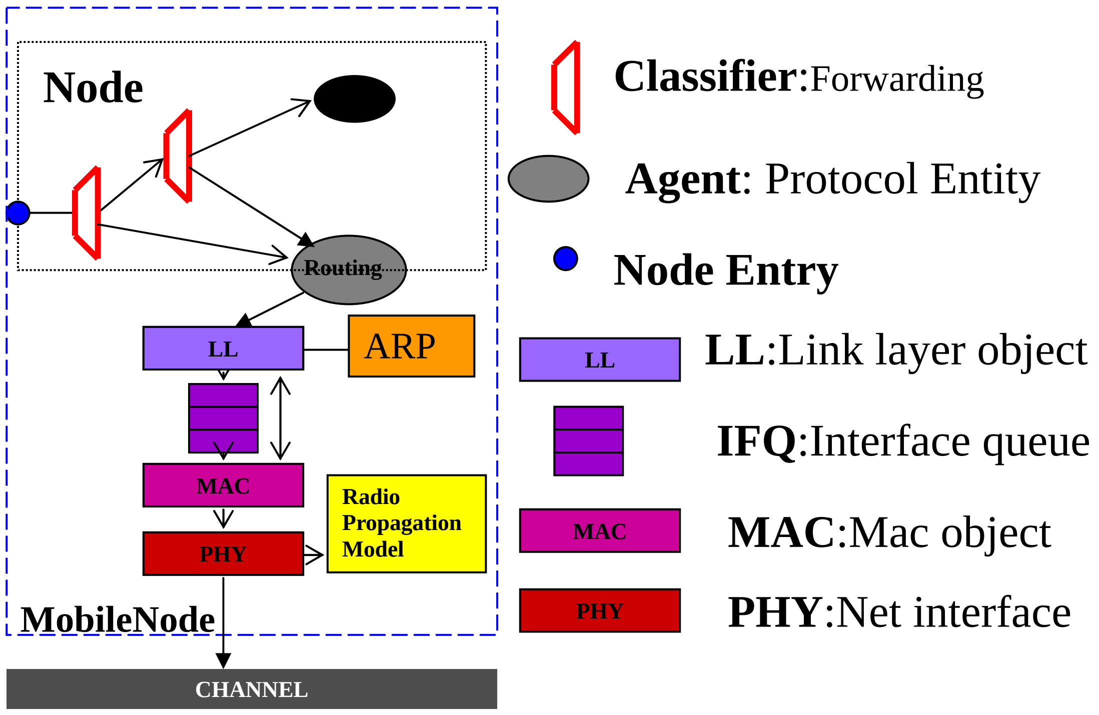
» C++ (Protocol/Layers)

» OTcl (Scenario/Topology)



All protocol specific implementations are done in C++, while testing scenarios are done in OTcl. Like those in original NS2 implementation, my new dual busy tone channels are also implemented in C++, which can communicate with other original components seamlessly.

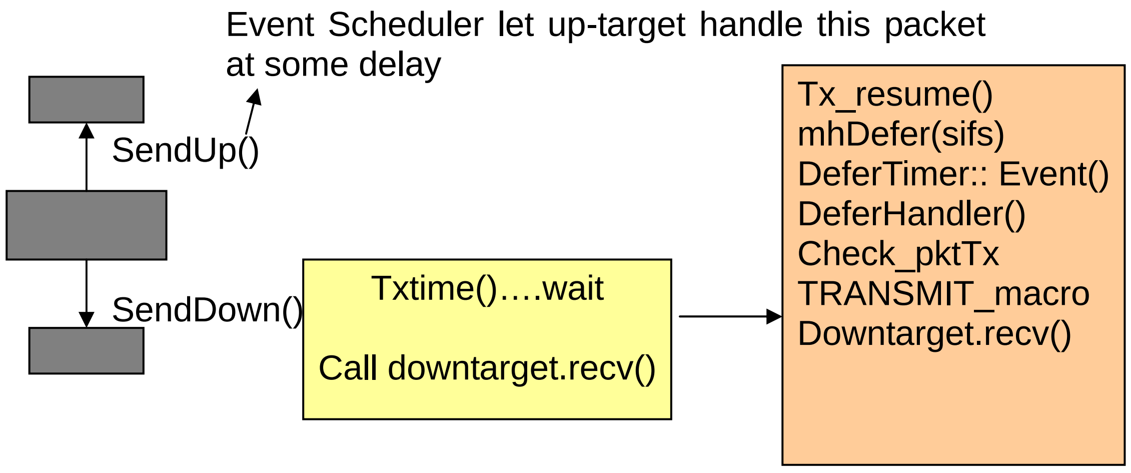
(2).Original Mobile Node Model in NS2



As show in the diagram, all physical channel information are held by MAC layer and PHY layer protocols. So, my implements work was done in MAC layer and PHY layer. One challenging part of this task is all the original NS2 protocols and interfaces have to be taken care of, otherwise new component won't work with rest part of NS2.

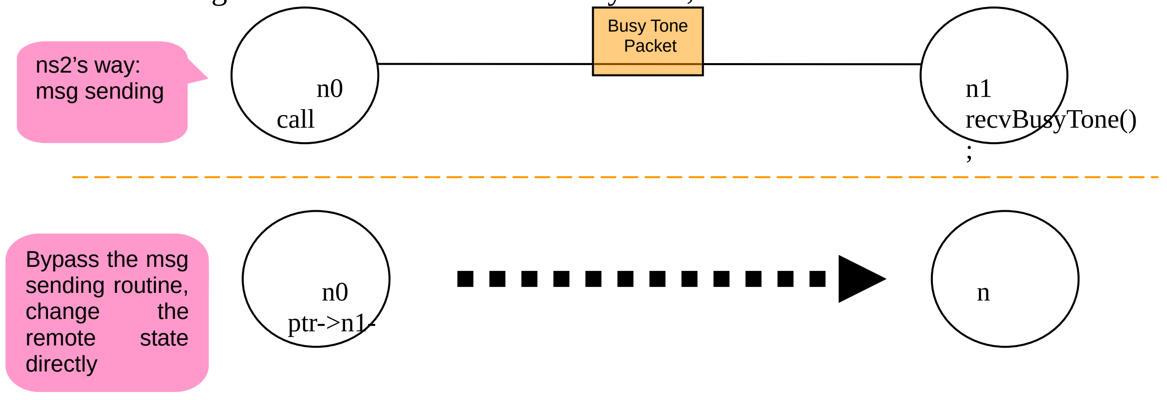
(3).My modifications into NS2

- Override recv() function
- Implement new state machine (adding internal member variable and functions), timers!
- Define new packet headers
- Event Scheduler

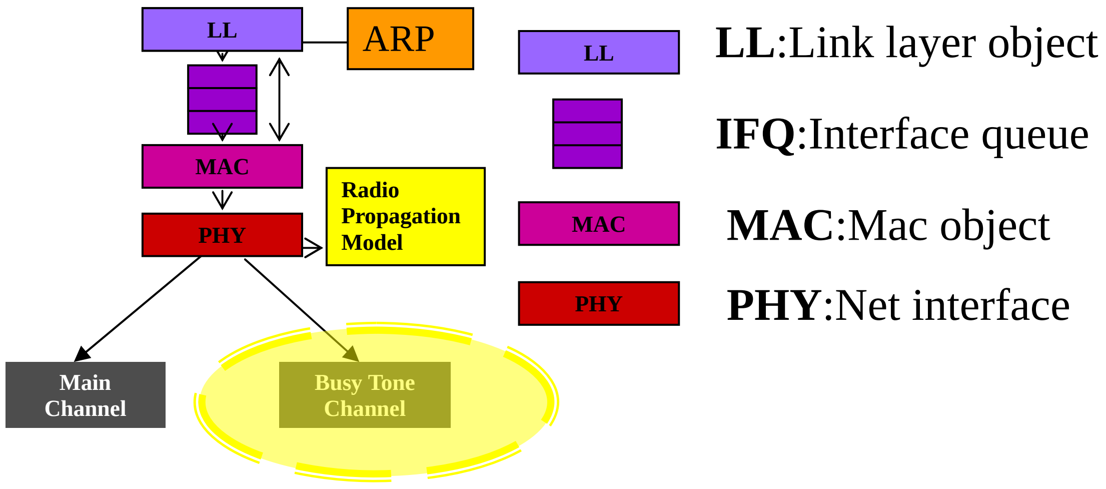


(4)Hacking into NS2 hierarchy

Given that essentially everything in ns2 is event, as long as the targeted nodes are be notified at the right time about the event of busy tone, it is done.



As shown in following diagram, logically an extra Busy Tone Channel should behave like an extra physical channel which could be used for sending and receiving busy tone signal.



2. Implementation Details:

(1). BusyTone class:

As mentioned in poster session, there are two extreme approaches for implementing extra busy tone for MAC protocols in ns2: hacking and extending ns2.

By hacking into MAC class instances, the whole design philosophy of ns2 would be violated. Originally in ns2, distributed nodes which are represented by multiple MAC class instances can only communicate with each other by sending and receiving packets. In the process of sending and receiving packets, calculations of propagation delay, transmission delay and power consumption can be done when downward classes (such as wireless physical channel) are passing the packets. The hacking method changes the state of remote nodes by using the pointers of the object instances to change the busy tone state variable directly. In this way, hacking method benefits in bypassing the complexity of class hierarchies of ns2, but suffers from losing the chance of simulating the wireless physical channel and power consumption.

By extending the original ns2 to support multiple interfaces and multiple channels, the implementation could benefit in:

- Embedding into existing architecture of ns2 seamlessly and gracefully.
- Having the calculation of various delays done by other classes.
- Reducing difficulties in implementing new multi-channel-multi-interface protocols.

However, the downside of the extending way comes together with its benefits: It requires diving into not only the C++ class hierarchies of ns2 but also the Tcl hierarchies, which could become very challenging and time consuming. In the C++ part of ns2, extending it to support multi-channel-multi-interface requires modifications of Mobile Node class, wireless-phy class, mac class … Almost the whole source code of wireless implementation in ns2 should be examined and modified whenever necessary. That's only part of the story. Due to the dual implementation of ns2 (C++ & Tcl), Tcl part requires corresponding modification if the C++ part is changed.

With all the above considerations and a tight schedule, the BusyTone class is implemented in a hybrid way of hacking and extending ns2.

(1). Storing pointers to BusyTone objects:

In initialization, BusyTone class object stores the pointers to itself into a global table. Save local BTMA object's pointer inside BusyTone. Whenever BTMA tries to send busy tone packets, BTMA objects call its own BusyTone object. One BusyTone object could access BusyTone instances of targeted node by looking up in the global table. In this way, BusyTone doesn't in fact send out or receive any packets using real channel.

(2).Asking GOD for neighbors:

In wireless simulation, there is a General Operation Director (GOD) which has access to location information of all the wireless nodes.

In the implementation of BusyTone, it utilizes the information from GOD to know which nodes are in the neighborhood.

(3). Calculating delays in BusyTone class:

BusyTone class is in fact a simplified MAC class which has all the delay calculations (processing delay, transmission delay and propagation delay) done by means of timers.

Implemented FSM in BTMA-NTS paper.

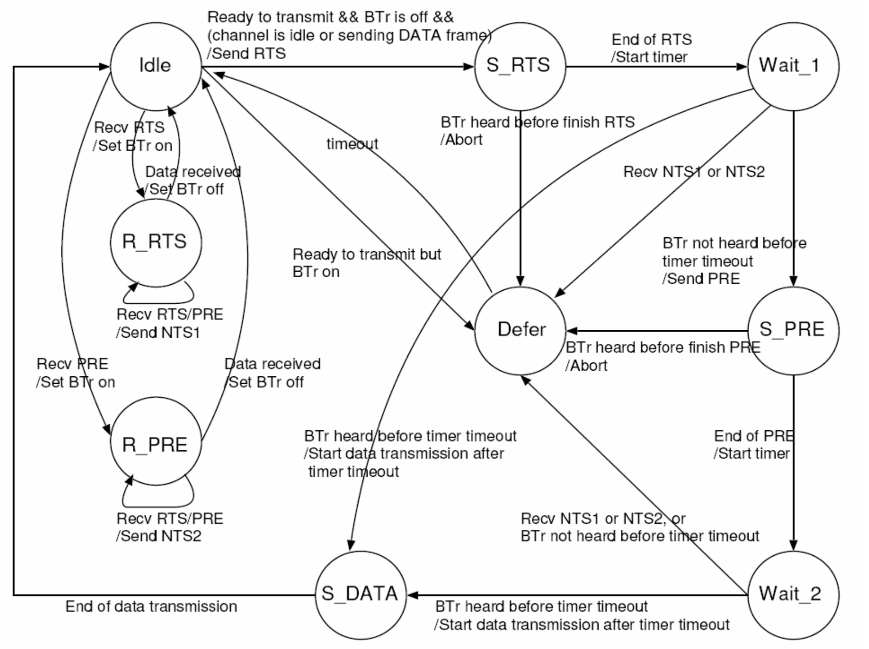
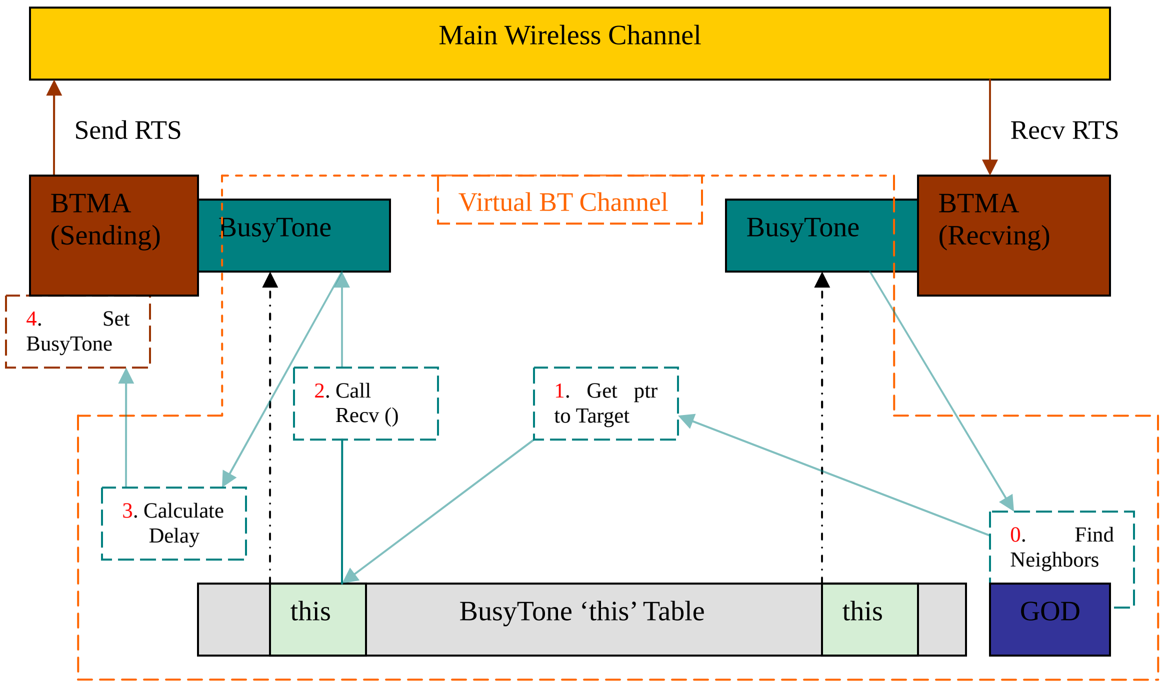


Illustration of implementation



Publication

Maggie Cheng, Yadi Ma, Yi Wang, "Improving Channel Throughput of WLANs and Ad Hoc Networks Using Explicit Denial of Request", IEEE Globecom 2006, pp1-6.