Technical Document on Wireless Virtualization

GDD-06-17

GENI: Global Environment for Network Innovations

September 15, 2006

Status: Draft (Version 1.0)

Note to the reader: this document is a work in progress and continues to evolve rapidly. Certain aspects of the GENI architecture are not yet addressed at all, and, for those aspects that are addressed here, a number of unresolved issues are identified in the text. Further, due to the active development and editing process, some portions of the document may be logically inconsistent with others.

This document is prepared by the Wireless Working Group.

Editors:

Sanjoy Paul, *Rutgers*

Srini Seshan, *CMU*

1 Introduction

This document is used to describe the collective thinking of the wireless sub-group of GENI to describe "virtualization" and "slicing" of wireless networks. The goal is to enable multiple experiments to share a common wireless network among multiple experiments that might either run concurrently or sequentially or in combination. Both long-running services as well as short-term experiments will be supported on this shared wireless network.

Definitions:

Slicing is the process of allocating a coherent subset (a slice) of GENI's physical resources to a specific experiment. Typically a slice will contain all of the resources needed to implement, or overlay a logical network on top of the GENI substrate.

- For example, an experimenter might implement a "virtual cache and forward" network by including in his slice a set of 802.11 Forwarding Nodes (FNs), a set of wireless links sufficient to interconnect the FNs, and a set of 802.11 Access Points (APs) connected to the wired backbone network.

Virtualization allows a single machine/node to implement multiple instances of a required logical resource within the same or different slices.

- For example, virtualizable 802.11 nodes could provide multiple experimenters with the illusion that each has access to and control over a separate 802.11 node at that location in the network.

Virtualization has been used extensively in wired network testbeds, such as PlanetLab, to efficiently support a large number of slices on the limited physical resources of the testbed. Unfortunately, the virtualization techniques (e.g., virtual machines, special CPU and bandwidth schedulers) that have been used in existing testbeds do not address the unique needs of wireless environments. The two key issues that wireless environments introduce are:

- **Isolation.** While resources are shared on existing wired testbeds, there are a number of effective techniques for ensuring that the resource usage of one experiment has little impact on others. Problems only appear when the testbed has insufficient resources (CPU, BW, etc.) to provide every experiment with a reasonable amount of resources. However, it is often possible to overprovision such testbeds. Unfortunately, overprovisioning cannot be applied to the key resource in wireless testbeds: the wireless spectrum. Due to the scarceness of wireless spectrum, wireless testbeds will need to accommodate a wider range or resource partitioning models to support a reasonable range of experiments.

- **Uniqueness of nodes.** Many testbeds are created from a collection of non-identical nodes. In such testbeds, experimenters may desire a particular collection of nodes either due to their unique properties or to recreate an earlier configuration. This issue is likely to be especially critical in wireless parts of GENI since wireless signal propagation: 1) is a

very node-specific property, 2) is difficult to control, and 3) has a significant impact on most experiments.

In the following, we describe how wireless network elements can be virtualized leading to efficient slicing of wireless networks for running both long-duration services as well as short-duration experiments.

2 Virtualization Techniques

Following techniques have been discussed for virtualization of wireless networks:

1) **Frequency Division Multiple Access (FDMA):** Virtualize a node by partitioning the frequencies.

   For example, an 802.11a node with 12 orthogonal frequencies can be logically partitioned into 4 virtual nodes by allocating 3 frequencies per virtual node as shown in Figure-1.

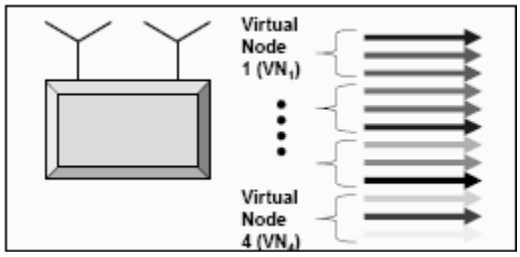

   Figure-1: FDMA-based Virtualization

   There can be as many as 12 virtual nodes in a single physical 802.11a node where each virtual node is allocated a single frequency. Although a node can be partitioned (into virtual nodes) along the frequency dimension, switching from one virtual node to another is not instantaneous. Channel switching time for Atheros cards is 5ms. while that for Intel cards is 20 ms. Because of the finite channel switching time (5 ms for Atheros cards for example), the virtual nodes will get their turn to transmit in a round robin manner over a cycle time of (5+x)*n ms where x= time for which a virtual node is active and n= # of virtual nodes. The duration over which a node is active (x ms) can be pre-configured and be made known to every node in the system.

   If multiple cards are used, channel switching time can be avoided but there may be some co-channel interference. There is also a context switching time (1-10 ms in Linux) which may or may not dominate when it's combined with channel switching.

   To summarize, FDMA refers to switching channels (or using multiple cards at different frequency partitions) within a physical node to emulate multiple virtual nodes.

2) **Time Division Multiple Access (TDMA):** Virtualize a node by partitioning along the time dimension. That is, different users get to use a given frequency partition in different "time slots". For example, an 802.11a node can be logically partitioned into 3 virtual

nodes by allocating 3 non-overlapping time slots to three experiments as shown in Figure-2.

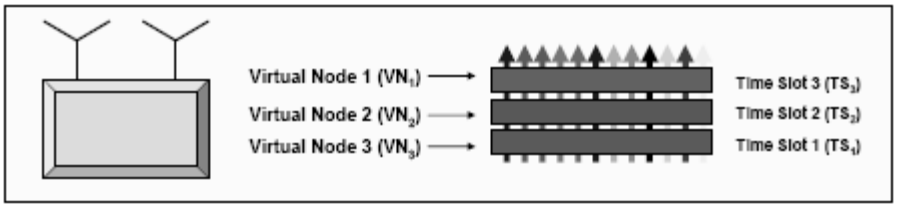

Figure-2: TDMA-based Virtualization

As the number of virtual nodes (obtained by partitioning in Time) increases per physical node, each experiment has to wait longer to get its turn to use the node.

Note that in this case there is a finite context switching time (1-10 ms in Linux) and hence the virtual nodes will get their turn to transmit in a round robin manner over a cycle time of (10+x)*n ms where x= the time for which a virtual node is active and n = # of virtual nodes

To summarize, TDMA refers to switching Time-slots within a physical node to emulate multiple virtual nodes.

3) **Combined FDMA and TDMA:** A node can be virtualized by allowing different users to use a given frequency partition in different "time slots" (TDMA).

For example, an 802.11a node with 12 orthogonal frequencies can be logically partitioned into 4 logical groups by allocating 3 frequencies per logical group and these logical groups can be further partitioned into multiple Time-slots as shown in Figure-3.. A given user will ALWAYS use the same frequency partition + time-slot combination. For example, virtual node 1 will always use [FP1,TS3].

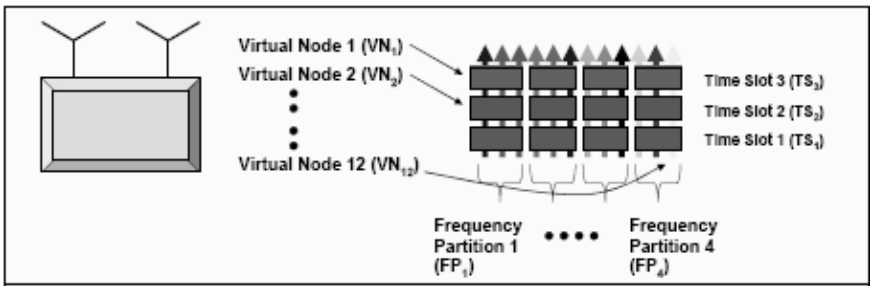

Figure-3: Combined FDMA and TDMA-based Virtualization

Although virtual nodes can be created by slicing a node along both frequency and time dimensions, switching from one virtual node to another is not instantaneous. Switching time in this case would consist of the channel switching time (5 ms for Atheros cards, 20ms for Intel cards), and the context switching time (1-10ms in Linux) from one process (running on a virtual node) to another. Therefore, the virtual nodes will get their turn to transmit in a round robin manner over a cycle time of (5+(x+10)*m)*n ms assuming Atheros cards and 10ms context switching time, where x = the time for which a user within a frequency partition is active, m = the number of users per frequency partition, and n = the number of frequency partitions. The duration over which a node is active (x ms) can be pre-configured and be made known to every node in the system.

To summarize, in combined FDMA+TDMA, a virtual node is identified by a unique COMBINATION of Frequency Partition and Time-slot.

4) **Frequency Hopping (FH):** A node can be virtualized by allowing different users to use different frequency partitions at different time-slots (Frequency Hopping - FH).

For example, an 802.11a node with 12 orthogonal frequencies can be logically partitioned into 4 logical groups by allocating 3 frequencies per logical group and these logical groups can be further partitioned into multiple time slots. However, unlike in combined FDMA+TDMA case, the SAME user will be using DIFFERENT frequency partition + time-slot combination in FH.

For instance, virtual node 1 (blue) will be using the following pattern [{FP1,TS3), {FP2,TS3},{FP3,TS2},{FP4,TS1}] in consecutive transmissions as shown in Figure-4.

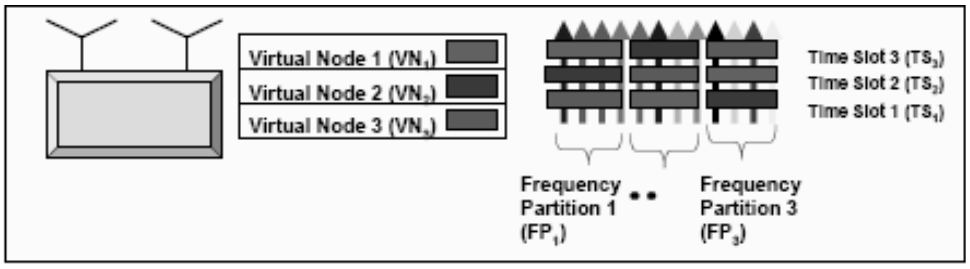

Figure-4: Frequency Hopping-based Virtualization

Note that FH will have scalability issues similar to FDMA+TDMA combination.

Thus a network "slice" for an experimenter would consist of a set of nodes in which the experimenter will use a pre-determined (frequency, time-slot) combination at a node in subsequent cycles where a cycle is defined as the time duration needed to complete transmission of all virtual nodes at a given physical node. This approach has been used in Virtual WiFi project of Microsoft Research.

To summarize, Frequency Hopping (FH) refers to partitioning of a node by allocating a unique SEQUENCE of Frequency and Time-slots to a virtual node

5) **Code Division Multiple Access (CDMA):** Virtualize a node by partitioning in "code" dimension. This applies to base stations that operate using orthogonal codes.

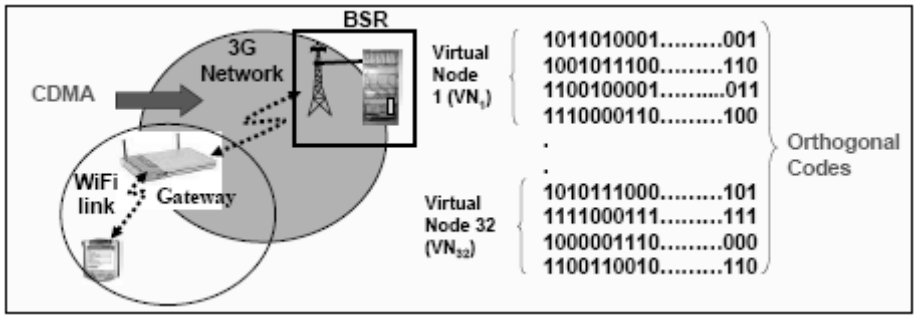

Figure-5: CDMA-based Virtualization

For example, a CDMA2000 1x node supports a maximum of 128 orthogonal codes. Thus, it can be partitioned into N virtual nodes by allocating M orthogonal codes per virtual node where NxM=128 as shown in Figure-5. In Figure-5, M=4 and N=32.

In a hybrid 3G-WiFi network as shown, a slice might consist of a set of orthogonal codes between 3G Base Station Router and Gateway; and a frequency partition between Gateway and end-user device (PDA/Laptop).

To summarize, CDMA refers to switching codes within a physical node to emulate multiple virtual nodes.

3 Slicing Techniques

Wireless networks and experiments pose several unique challenges that make "slicing" different for wireless networks compared to wired network. First, many wireless experiments focus on new Physical Layer and MAC layer protocols, and hence sharing a single "physical" wireless node between multiple such experiments becomes impossible. Second, mobility and handoff experiments typically have very tight latency requirements (~ ms) which also become extremely difficult, if not impossible, to support on a network where the experimenter gets access to "less than full" resources of the set of physical nodes in the topology of interest. Emerging applications, such as Vehicular networking, also have very stringent latency requirements mainly because the time to detect impending collision and react to it would be in the order of 10ms which may not be achievable with an abstraction of a "virtual" node in a shared wireless network.

An over-arching criterion that makes wireless networks fundamentally different from wired network is the concept of "topology" in wireless experiments. In fact, all nodes are not equal in wireless experiments, and wireless experiments are very much topology-dependent because of

RF. For instance, a VoIP over wireless experiment with 1 AP and 2 laptops near the AP would give very different result compared to one with 1 AP and 2 laptops far from the AP. In these cases, it's not enough to virtualize nodes and create slices using virtualized nodes. The notion of topology needs to be incorporated in the slice allocated to the experimenter.

First we show how by using a combination of the virtualization techniques enumerated above, wireless networks can be "sliced" in a variety of ways, and then discuss a possible way to handle topologies:

1) **Space Division Multiple Access (SDMA):** In this technique, a FULL node is allocated to a given user and no virtualization is done. For example, on a grid-like topology, 802.11 a/b/g nodes would be partitioned such that multiple users can simultaneously use the grid and run experiments in their respective partition. Partitioning is done using "spatial" separation so that nodes within a partition do not interfere with nodes in a different partition as shown in Figure-6. SDMA partitioning is limited by physical span for a given transmit power.

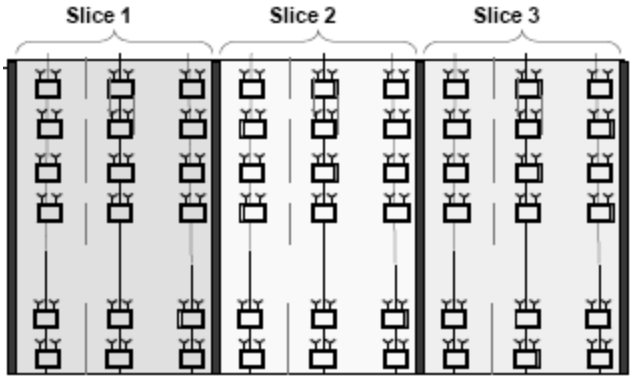

Figure-6: SDMA-based Slicing

In some testbeds, such as ORBIT, where the wireless nodes are in the physical proximity of one another, it is not possible to partition the nodes via a physical separation. In such cases, the technique of "artificial stretching" can be used to logically stretch the space that houses the nodes. Artificial stretching is achieved by controlling the "transmit power" of the nodes and using "noise injection" to put up artificial barriers between the nodes belonging to different partitions..

Note that in SDMA, each node is "entirely" assigned to a network "slice". This is in contrast to FDMA, TDMA or FH where a "logical" partition of a physical node (referred to as a "virtual" node) is assigned to a network "slice". Essentially SDMA enables simultaneous use of a wireless network by partitioning the network and allowing each experimenter to run her experiments in her assigned set of nodes (which is her "slice").

2) **Combined SDMA and TDMA:** In this technique, a grid of 802.11 a/b/g nodes is partitioned using "spatial" separation, and the "spatial" slices are further partitioned in the time dimension by creating "time slots". For example, there are 3 spatial partitions in

Figure-7, and within each spatial partition, there are three time slots. Thus 3x3 = 9 simultaneous experiments can share the grid.

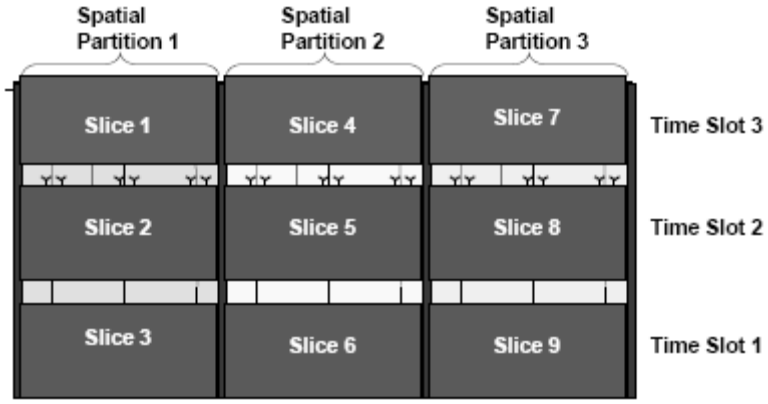

Figure-7: Combined SDMA and TDMA-based Slicing

3) **Combined SDMA and FDMA:** In this technique, a grid of 802.11 a/b/g nodes is partitioned using "spatial" separation, and the "spatial" slices are further partitioned in the frequency domain by creating "frequency partitions".

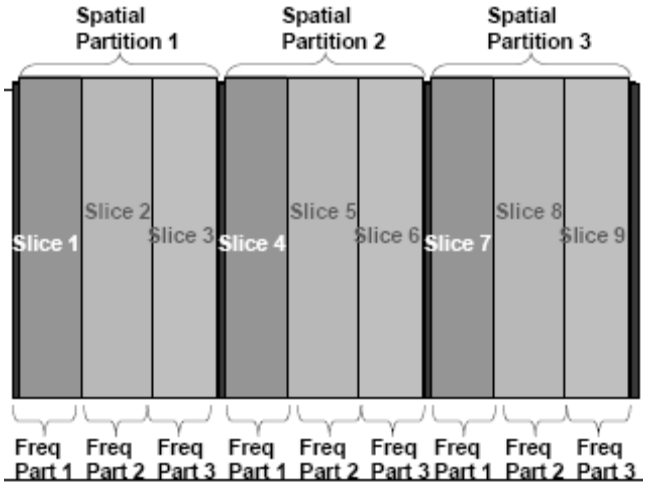

Figure-8: Combined SDMA and FDMA-based Slicing

For example, there are 3 spatial partitions in Figure-8, and within each spatial partition, there are three frequency partitions. Thus 3x3 = 9 simultaneous experiments can share the grid.

4) Combined SDMA, FDMA and TDMA: In this technique, a grid of 802.11 a/b/g nodes is partitioned using "spatial" separation, and the "spatial" slices are further partitioned in

the frequency domain by creating "frequency partitions" which in turn are partitioned in time domain by creating "time slots".

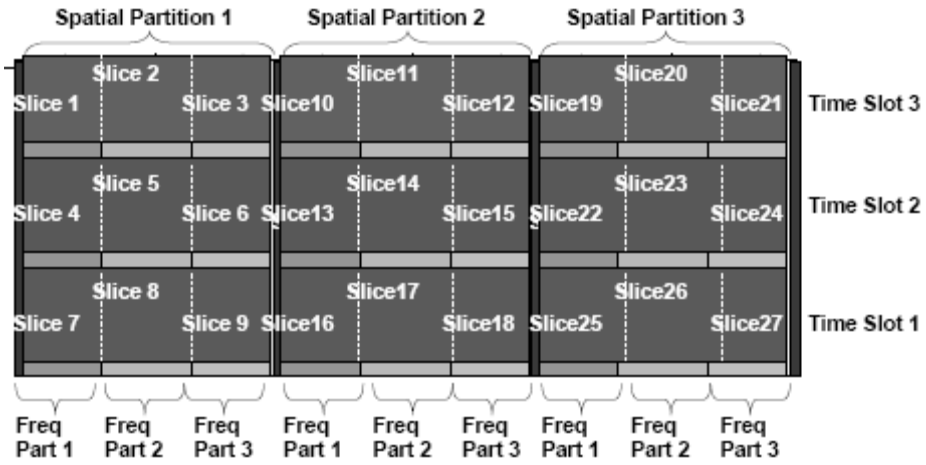

Figure-9: Combined SDMA, FDMA and TDMA-based Slicing

For example, there are 3 spatial partitions in Figure-9, and within each spatial partition, there are three frequency partitions, and 3 time slots. Thus 3x3x3 = 27 simultaneous experiments can share the grid.

A discussion on handling topologies in "virtualized" wireless networks

Assuming SDMA+FDMA slicing, essentially each point (node) in a grid of nodes can be represented by an n-tuple $[F_1, F_2, \ldots, F_n]$ where F_i, i=1,.., n is the i^{th} frequency partition at each node and $F_i = 1$ or 0 depending on whether F_i is available or not.

Greedy Algorithm:

1. As wireless topologies are allotted to experiments, status of corresponding F_i's becomes 0 at a node that is part of the topology.

2. As experiments complete, the corresponding F_i's are reclaimed

3. Requested topology is assigned as long as there are enough virtual nodes to satisfy the requirement.

Drawbacks of the Greedy algorithm: If the Greedy algorithm is run continuously, it results in sub-optimal allocation of topologies; and hence the resources of a wireless network may not be fully efficiently utilized. In order to better utilize the shared wireless resources, it might require periodic flushing of the experiments, and reset the network to make all F_i 's available at each node.

Optimal Algorithm: Without going into the details of the optimal algorithm, it is clear that there would be a need to re-evaluate existing slices and re-allocate resources as and when a new request arrives. However, that may not be a good idea because it might require storing and restoring a lot of state information, making the system unscalable.

In fact, preliminary experience suggests a **Hybrid Algorithm** between Optimal and Greedy would be most practical.

Currently, in state-of-the-art shared wireless testbeds, such as ORBIT, a variety of topologies are allowed for wireless experiments. However, since there is only ONE user at a given time in the wireless grid, allocation of topology is easier than when multiple users run experiments on the same grid concurrently.

4 Mapping Wireless Networks to Virtualization Techniques

All virtualization techniques are not equally amenable to all types of wireless networks. Specifically, we consider five different types of wireless networks:

1. Emulator networks: these kinds of wireless networks are typically deployed in "controlled" environments such that experiments can be "repeated" with predictable results. Examples of "emulator" networks are ORBIT [ ], Whynet [ ], Emulab [ ] etc. ORBIT, for example, is a grid of 400 wireless nodes with multiple wireless interfaces per node where experimenters can remotely download their own code, run experiments, monitor the behavior of the network, and collect results.

2. Urban Grid: these wireless networks consist of large number of 802.11a/b/g nodes deployed in a mesh network covering a downtown area such that anyone with a laptop/PDA equipped with 802.11a/b/g card would be able to communicate with rest of the world. WiFi Mesh networks covering entire downtown Philadelphia or Palo Alto would be examples of urban grid.

3. Suburban Hybrid: covering a large geographic expanse of suburbia with short-range radios, such as 802.11a/b/g would not be economical, while cellular networks are meant to cover these areas. In general, it makes sense to complement cellular networks with 802.11a/b/g networks in hot spots to provide the most economical network with maximum bandwidth.

4. Cognitive Radio networks: software defined radios with flexible frequency range tuning are challenging the conventional hardwired radios that are engineered to operate within their designed frequency bands. These radios, despite being in the very early stages of development, will be included as a distinct type in the mix of wireless networks under consideration.

5. Sensor networks: most of the sensor networks that are deployed in the field have very "specific" functionality. However, there are some "generic" sensor platforms (such as,

Mica II mote) that can be leveraged for multiple sensor experiments. Our goal is to explore how a testbed of such sensors can be shared across multiple experimenters.

The following table summarizes the mapping between wireless networks and the virtualization techniques.

	SDMA	SDMA+ TDMA	SDMA+ FDMA	SDMA+ FDMA+ TDMA	SDMA+ FH	SDMA+ CDMA
Emulator	Yes	Yes	Yes	Yes	Yes	No
Urban Grid	Yes (all slices not equal)	Yes (needs tight time sync)	Yes	Yes (needs tight time sync)	Yes (needs tight time sync)	No
Suburban Hybrid	Yes	Yes	No (Tech dependent)	No (Tech dependent)	No (Tech dependent)	Yes
Cognitive Radio	Yes	No	No	No	No	No
Sensors	Yes	Potentially "yes", but practically "no" (limited CPU/ memory)	Potentially "yes", but practically "no" (limited CPU/ memory)	Potentially "yes", but practically "no" (limited CPU/ memory)	Potentially "yes", but practically "no" (limited CPU/ memory)	No

Essentially, urban grids and emulator networks can be virtualized in the same way (FDMA, SDMA, TDMA, FH) except that urban grids are subject to "uncontrolled" external interferences while the emulator networks are not. As a result the likelihood of being able to provide a "guaranteed" quality of service for a slice of the network is much smaller for urban grids compared to that for emulator networks. In addition, for SDMA, all partitions (slices) would be pretty similar for emulator networks, but that is not the case for urban grids, because the characteristic of the slice on the urban grid would depend on a number of factors, such as, the density of uncontrolled users with laptop/PDA accessing the nodes in the partition, the terrain in which the partition is situated, and other external factors that are usually not relevant for emulator networks.

In case of suburban hybrid networks, SDMA can be used to slice the networks because one or base stations (from cellular network) can be assigned together with a set of 802.11a/b/g nodes to an experimenter. In addition, TDMA overlay can also be used to slice such networks. However, FDMA or FH may not be feasible because the basic wireless technology used in cellular (CDMA2000, W-CDMA) may not be amenable to slicing by frequency.

In theory, a cognitive radio node is amenable to a number of different virtualization techniques. However, there are two factors that limit the practicality of these techniques. First, most cognitive radio radios are extremely resource limited in terms of the processing available on the actual interface, the bandwidth available between the interface and host CPU and the processing available on the host CPU. In addition, many of the resources on the cognitive radio interface do not provide a "virtualized" interface. The techniques that we discuss in this document largely focus on virtualizing spectrum usage and do not address virtualizing this interface. The second

complication is that many of the experiments that use cognitive radios will expect significant flexibility in managing spectrum from themselves – as this is often the goal of cognitive radio systems. As a result, TDMA, CDMA and FDMA schemes may be impractical for such experiments.

Regarding sensor networks, if the sensors are used for a specific function, the network can be used by one experimenter at a time or if the network is large enough, it might be partitioned physically (SDMA) to allow multiple experiments. In case the sensor platform is generic, the network can be partitioned either along the frequency dimension or along the time dimension or by using a combination in the frequency-time grid. For example, the Mica II motes have 32 orthogonal frequencies and the switching time between frequencies is on the order of 250 s, and thus we might leverage FDMA virtualization on such platforms. However, the platform may not have enough processing power/memory to be able to support more than one application running at the same time. Thus SDMA seems to be the right virtualization technique for sensor networks.

5 Mapping Wireless Applications/Experiments to Virtualization Techniques

Just as different wireless networks have different characteristics, different applications also have different requirements, and hence all experiments may not be supported by all types of virtualization. In addition, experimenters may need to run their experiments over different durations of time. Specifically, there are three broad classes of experiments:

a. Short-term (< 1 day)

b. Medium-term (< 1 month: few weeks)

c. Long-term (> 1 month: several months)

The mix of experiments that can be supported on GENI Wireless network will be a scheduling problem taking into account the “resource requirements" of the experiments and the “resource availability” in the network. The goal of scheduling would be to maximize the number of experiments while being fair to experimenters with long-term experiments. This is where “virtualization” of wireless networks will intersect with the “control and management” plane of GENI.

As mentioned earlier, specific experiments would have specific requirements. For example, the following is a partial list of “experiment-specific” constraints:

a. Very small delay/jitter bound (< ms)

b. Small-to-Medium delay/jitter bound (VoIP/Conferencing: ~100ms)

c. Low Packet Loss (Control/Alert Message delivery)

d. High Throughput (P2P file delivery)

e. Mobility: Programmed and Natural

A discussion on TDMA overlay is in order before indicating the mapping of types of experiments to virtualization techniques. When TDMA is used as an "overlay" on existing MAC rather than an "underlay" for the MAC itself, there are some limitations of a scheme that uses TDMA-overlay-based virtualization. A TDMA "overlay" Time Slot consists of three components: (1) Channel Acquisition time (includes the time for backoffs in CSMA/CA), (2) Packet Transmission time (includes the time a node is active), and (3) ACK time (includes the MAC-level ACK). Assuming 802.11a/g @24 Mbps, 1KB packet transmission/node, 24B ACK, and a single backoff, the Channel Acquisition time is approximately 500 s, Packet Transmission time is approximately 350 s and ACK time is approximately 8 s, thereby making a TDMA overlay time-slot equal to 858 s or roughly 1 ms. If 802.11b nodes are used instead of 802.11a/g, the available bandwidth would be 1 Mbps instead of 24 Mbps, and hence the time-slot will be approximately 10 ms. In addition, when time slots are switched between processes running on the same node and same frequency, there is a "context" switching time on the order of 10 ms. If in addition, frequency channel is switched, that would cost an additional 5-20 ms depending on the type of 802.11a/b/g card on the node. Adding up these times, an experimenter, sharing a wireless network with another experimenter, would be able to access the network every 25-40 ms. If there are N>2 experimenters, the time between consecutive TDMA slots would be (N-1) times 25-40 ms. These numbers clearly show the limitation of TDMA-overlay-based virtualization technique for experiments that have very small (order of 1-10 ms) delay/jitter requirements.

Another topic that requires special discussion encompasses MAC layer and Cross-Layer wireless experimentation. Sample MAC experiments that can be performed in SDMA, and SDMA+FDMA slices are:

- ✓ Resource control (Rate, Power adjustments)
- ✓ Admission control
- ✓ QoS control, such as, changing priority, adjusting # of MAC retransmissions
- ✓ Adjusting other MAC-level parameters, such as, Carrier-Sense Thresholds in CSMA/CA, Disabling of Acks, Disabling of back-off timers

Cross-Layer experiments involving MAC layer and above can be performed in SDMA, and SDMA+FDMA slices. Experiments involving Frequency control cannot be performed in SDMA+FDMA slices, but can be performed in SDMA slices. In order to perform two simultaneous cross-layer experiments on the same network, SDMA would be the best slicing technique. However, SDMA+FDMA slice can also be used provided the experiments do not have very tight (~10ms) "latency/jitter" requirements. This is because context switching between cross-layer experiments would involve storing and restoring a lot of protocol state at each node

The following table captures the mapping of types of experiments to various virtualization techniques.

	SDMA	SDMA+ TDMA	SDMA+ FDMA	SDMA+ FDMA+ TDMA	SDMA+ FH	SDMA+ CDMA
MAC Experiments	Yes	Yes (limited)	Yes (limited)	Yes (limited)	Yes (limited)	Yes (limited)
Very Small delay/jitter (< ms)	Yes	No	No	No	No	No
Small-to-Medium delay/jitter (~ 100ms)	Yes	Yes (provided # of slices is limited)	Yes (provided # of slices is limited)	Yes (provided # of slices is limited)	Yes (provided # of slices is limited)	Yes
Low Loss	Yes	Yes	Yes	Yes	Yes	Yes
High Thruput	Yes	Yes (depends on # of experiments sharing resources and the desired throughput)	Yes (depends on # of experiments sharing resources and the desired throughput)	Yes (depends on # of experiments sharing resources and the desired throughput)	Yes (depends on # of experiments sharing resources and the desired throughput)	Yes

In addition to the delay/loss/throughput requirements of the experiments, some experiments may be related to assignment of channel on the wireless networks, or may involve mobility and handoff. Mobility can be either "programmed" as in the case of a rover roaming around in an area covered by wireless nodes or "natural" as in the case of pedestrians walking around in a downtown covered by 802.11a/b/g nodes. Programmed mobility is best handled by SDMA while natural mobility can be handled by any one of FDMA, SDMA, TDMA of FH. However, if the user wants to experiment with handoff for time-sensitive applications, such as, VoIP, the limitations of TDMA/FDMA/FH as mentioned before would apply.

	SDMA	SDMA+ TDMA	SDMA+ FDMA	SDMA+ FDMA+ TDMA	SDMA+ FH	SDMA+ CDMA
Mobility: Programmed	Yes	Yes (rover has its own space)	No	No	No	Yes
Mobility: Natural	Yes	Yes	Yes	Yes	Yes	Yes
Channel Switching	Yes	Yes	Limited	Limited	Limited	Limited
Multicast	Yes	Yes	Yes	Yes	Yes	Yes

6 Scalability and Limitations

While some of the limitations of virtualization techniques have been discussed in earlier sections, this section will contain a comprehensive discussion on scalability issues.

1) **FDMA:** Scalability is limited by:
    a. Number of Orthogonal frequencies (3 for 802.11b, 12 for 802.11a/g)

   b. Mix of radio nodes (802.11a/b/g, Bluetooth, Mica-II motes)
   c. Switching time between frequencies (5 ms Atheros, 20ms for Intel)
   d. Active Time for each virtual node (# transmitted packets/time slice)
   e. Multiple card-based approach will be subject to some co-channel interference
   f. Mix of experiments (with varying throughput/latency needs)
2) **SDMA:** Scalability is limited by:
   a. Number of "non-interfering" partitions
   b. Number of nodes in each partition
3) **TDMA Overlay:** Scalability is limited by:
   a. Switching time between frequencies (5 ms for Atheros, 20 ms for Intel)
   b. Switching time between time-slices within a given frequency (context switching time: 1-10ms if no disk access; 100 ms if disk access is involved)
   c. Channel Acquisition time (assuming existing MAC, such as CSMA/CA)
   d. Active time for each virtual node (# transmitted packets/time slice)
   e. Mix of experiments (with varying throughput/latency needs)
4) **Frequency Hopping:** Scalability is limited by:
   a. Number of Orthogonal frequencies (3 for 802.11b, 12 for 802.11a/g)
   b. Switching time between frequencies (5 ms Atheros, 20ms for Intel)
   c. Switching time between time slices within a given frequency (context switching time: 1-10ms if no disk access; 100 ms if disk access is involved)
   d. Channel Acquisition time (assuming existing MAC, such as CSMA/CA)
   e. Active time for each virtual node (# transmitted packets/time slice)
   f. Mix of experiments (with varying throughput/latency needs)
5) **Artificial Stretching:** Scalability is limited by:
   a. Number of "non-interfering" partitions
   b. Granularity of control on range of Noise source
   c. Granularity of control on transmit/receive power of nodes
   d. Number of nodes in each partition

7 Recommendations

1. **Emulator networks**: Such networks should use a combination of SDMA+FDMA+TDMA+artificial stretching. The hardware used in these testbeds should be chosen to accommodate this full combination. However, it is likely that the FDMA & TDMA part of the combination cannot support some experiments. As a result, these testbeds should employ scheduling algorithms that operating different spatial slices in different modes, such that the current experiment mix can be supported efficiently.

2. **Urban Grid**: Such networks should support SDMA+FDMA+TDMA – artificial stretching is unlikely to be useful in this setting. In addition, since these testbeds target "realistic" (real-time, real-usage patterns and real propagation propeties) workloads, it is unlikely that many of the experiments that operate on such testbeds would be able to accommodate FDMA or TDMA. As a result, the testbed should be sized and operated

assuming SDMA is the primary form of virtualization while keeping provisions for limited use of FDMA and TDMA-based slicing.

3. **Suburban Hybrid**: These networks certainly can be used to support SDMA-based slicing because that is the easiest way to isolate experiments. However, depending on the technology deployed, other forms of slicing can also be deployed. For example, if CDMA-based networks are deployed, CDMA-based slicing is highly feasible on top of SDMA-based slicing. In addition, there should be provisions for limited form of TDMA-based slicing that can be combined with SDMA and CDMA-based slicings to run certain types of experiments. On the other hand, if OFDM-based systems are deployed instead of CDMA-based systems, FDMA-based slicing can be used instead of CDMA-based slicing.

4. **Cognitive Radio**: These should largely use SDMA for slicing. This implies certain requirements for ensuring that there a sufficient number of slices can be supported. First, there nodes need to span a sufficiently large physical area such that a number of slices can be created. Second, careful scheduling algorithms may need to be developed to ensure that different experiments can be supported in a timely fashion. Third, efforts should to develop techniques to virtualize the cognitive radio hardware should be encouraged. This will enable more efficient slicing of cognitive radio testbeds.

5. **Sensor networks:** These should largely use SDMA for slicing. This implies certain requirements for ensuring that there a sufficient number of slices can be supported. First, there nodes need to span a sufficiently large physical area such that a number of slices can be created. Second, careful scheduling algorithms may need to be developed to ensure that different experiments can be supported in a timely fashion.