CRL Distribution in VANETs using ns-3

Michael E. Nowatkowski

Workshop on ns-3

Malaga, Spain

15 March 2010

nsa Network Security and Architecture Laboratory

Georgia Tech College of Engineering
School of Electrical and Computer Engineering

Agenda

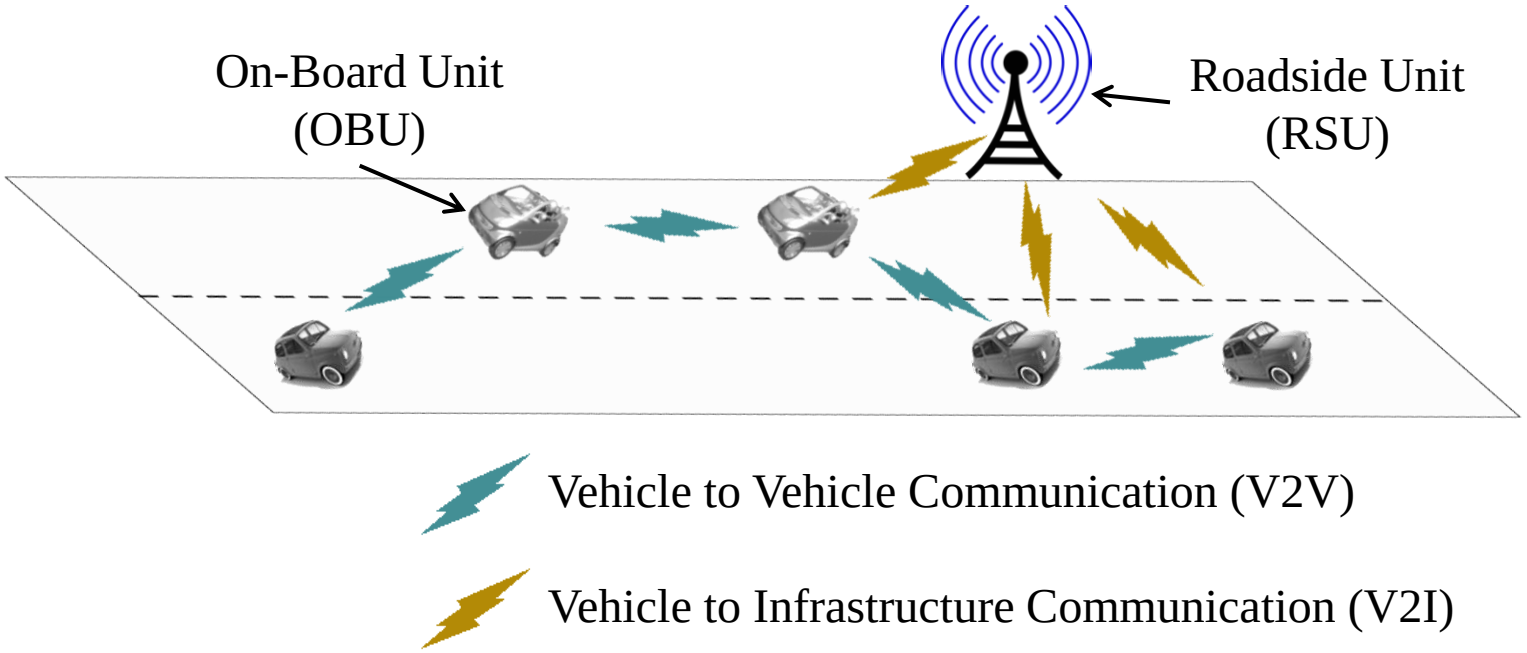
- VANET Overview
  - Physical Layer
  - MAC Layer
  - Network Layer
  - Application Layer
- CRL Distribution Methods
  - Most Pieces Broadcast
  - Generation per Channel
- Mobility Traces

Background

- Not a programmer
- Using ns-3-dev, somewhere between 3.6 and 3.7
  - changeset: 5834:8481618b577f
  - date: Thu Dec 03 10:54:05 2009 +0300
- Working on distributing files in a VANET environment

Vehicular Ad Hoc Network Overview

- Network-enabled vehicles and infrastructure
- IEEE Trial-use Standard 1609 (4 parts)
- Increased safety and convenience



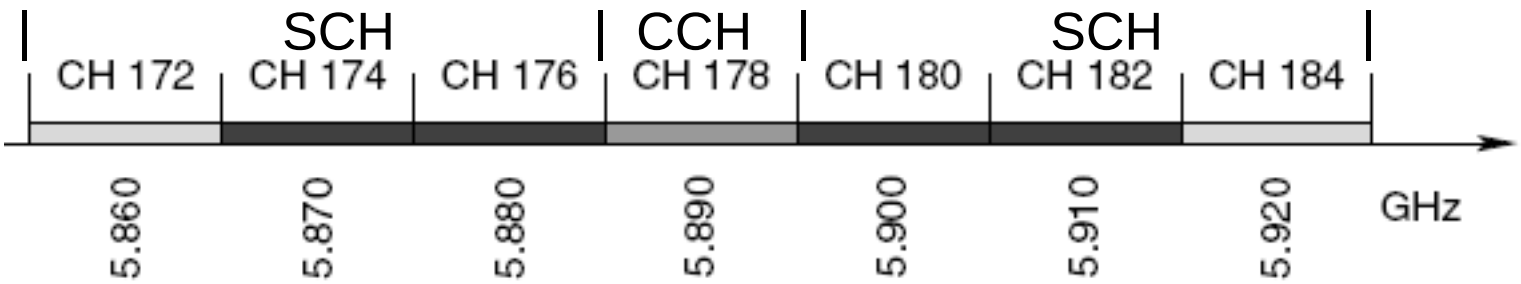
Increased Safety and Convenience

- Safety
  - Emergency electronic brake light
  - Pre-crash sensing
  - Cooperative forward collision warning
  - Traffic signal violation warning
  - Curve speed warning
  - Lane-change warning
  - Left turn assistant
  - Stop sign movement assistant
- Transportation efficiency
  - Traffic re-routing
  - Enhanced guidance and navigation
  - Lane merging assistant
- Information and entertainment
  - Mobile Internet
  - Point of interest (POI) notification
  - Fuel consumption management
  - Vehicle diagnostics

Source: Vehicle Safety Communications Consortium, 2006

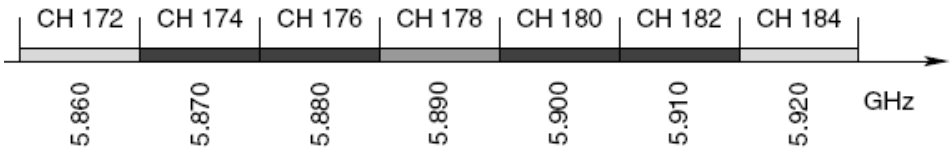
Physical Layer

- IEEE 802.11p is the physical-layer protocol
- Seven 10 MHz channels in the 5.9 GHz range
  - Data rates of 3 to 27 megabits per second
  - One control channel (CCH) → beacons and safety messages
  - Six service channels (SCH) → data exchange

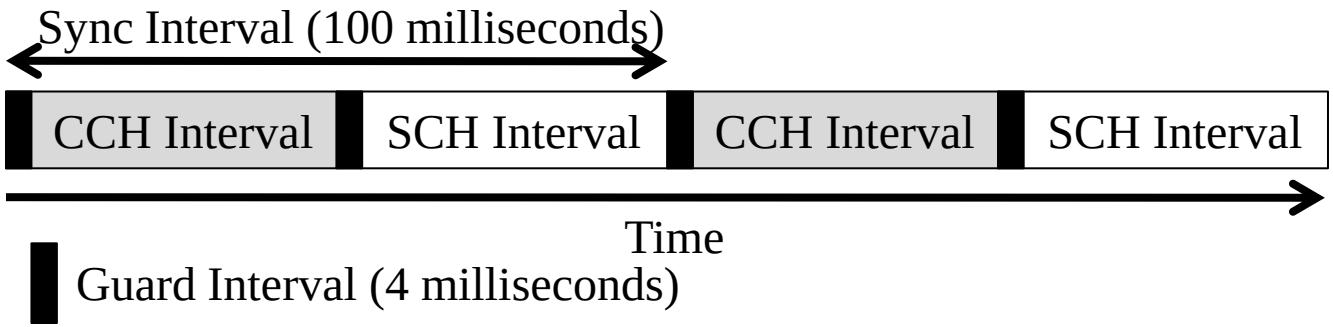


Physical Layer

- IEEE 802.11p is the physical-layer protocol
- Seven 10 MHz channels in the 5.9 GHz range
  - Data rates of 3 to 27 megabits per second
  - One control channel(CCH)
  - Six service channels(SCH)



- Nodes monitor the CCH periodically(every 100 ms)



Node Creation

- Helpers:

```
WifiHelper wifiHelper = WifiHelper::Default ();
QosWifiMacHelper macHelper = QosWifiMacHelper::Default ();
macHelper.SetType ("ns3::QadhocWifiMac");
YansWifiPhyHelper phyHelper = YansWifiPhyHelper::Default ();
YansWifiChannelHelper chHelper = YansWifiChannelHelper::Default ();
```

- Find dev for channel switching:

```
Ptr<WifiNetDevice> dev =
  DynamicCast<WifiNetDevice>((*nc_RSU.Get(i)).GetDevice(0));
Ptr<WifiPhy> phy = dev->GetPhy();
```

(ChannelNumber(uint16_t) is now an attribute of YansWifiPhy)

Physical Layer

- Set up Wifi parameters for 300 meter range

```
phyHelper.Set("EnergyDetectionThreshold", DoubleValue(-96.0));
phyHelper.Set("CcaMode1Threshold", DoubleValue(-99.0));
phyHelper.Set("RxNoiseFigure", DoubleValue(4));
```

- Set up Control Channel and Service Channel intervals

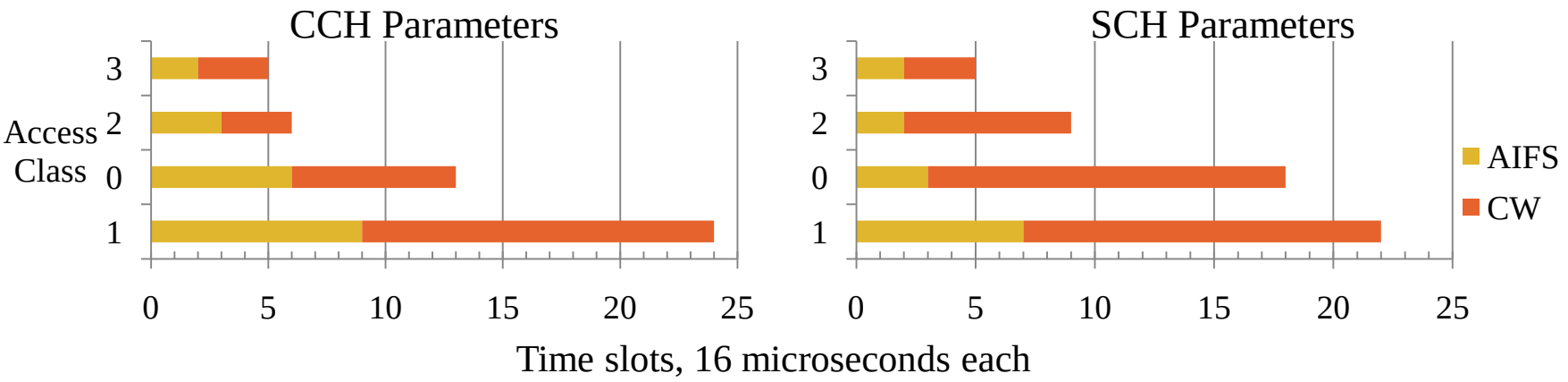
```
Simulator::Schedule(Seconds(0.0), &SwitchChannel, phy, 178, cchRSU);
  // Control Channel
Simulator::Schedule(Seconds(syncInterval/2), &SwitchChannel, phy,
  174, schRSU); // Service Channel

Void SwitchChannel…
Simulator::Schedule(Seconds(syncInterval), &SwitchChannel, phy,
  channelNumber, p_sch);
```

MAC Layer

- 802.11p uses User Priority, similar to 802.11e
- CCH and SCH parameters are slightly different

```
wifiHelper.SetStandard(WIFI_PHY_STANDARD_80211p_SCH);
wifiHelper.SetStandard(WIFI_PHY_STANDARD_80211p_CCH);
```

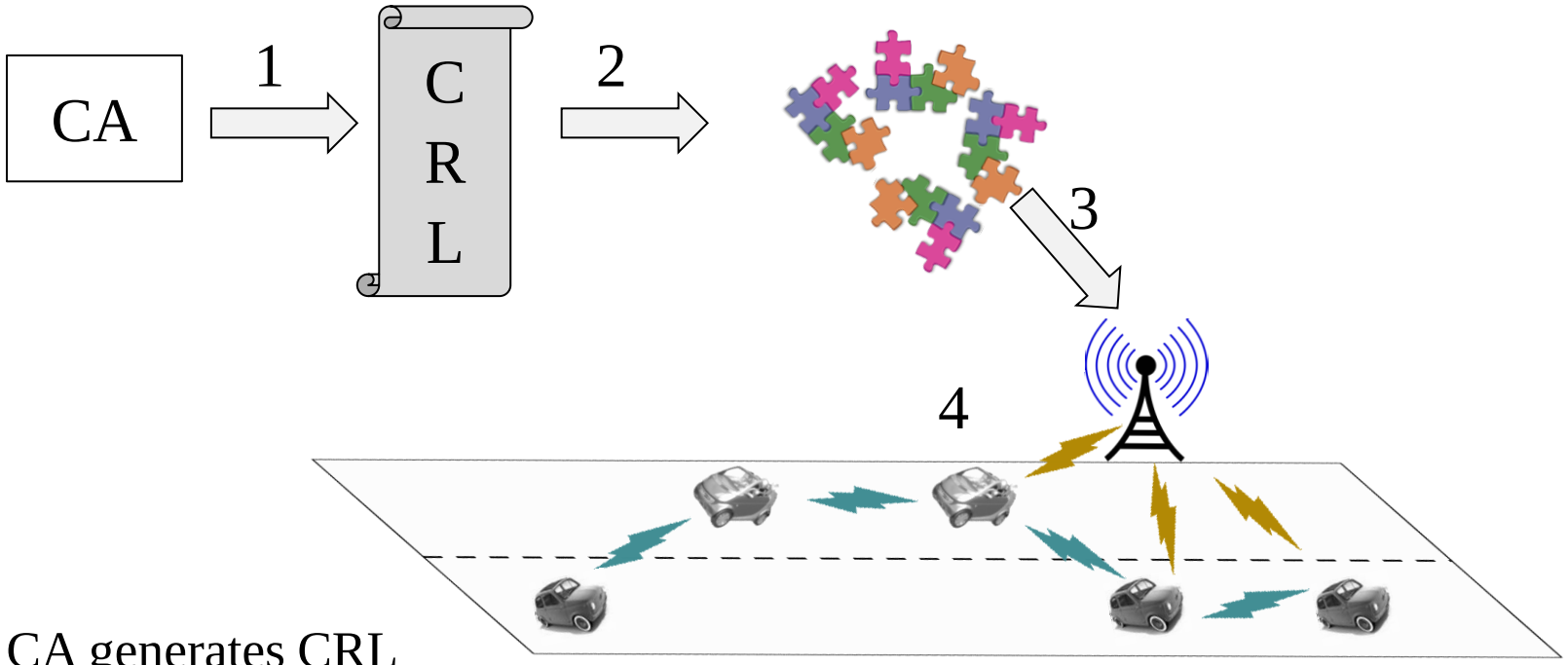


- Broadcast used for beacons and CRL pieces

Network Layer

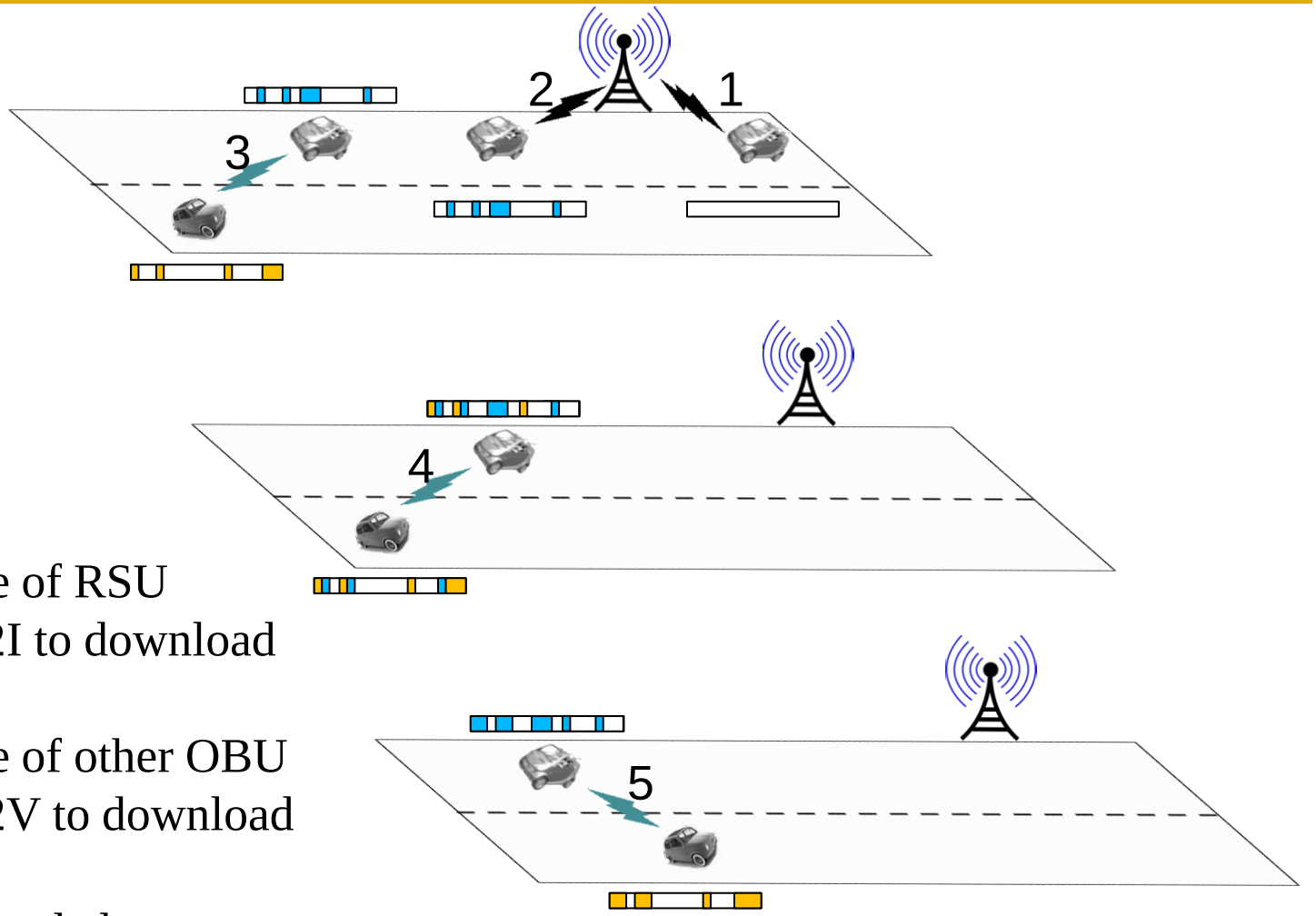
- 802.11p supports WSMP and IPv6
  - WSMP allowed on both Control Channel and Service Channels
  - IPv6 allowed on Service Channels
- WSMP does not use MAC address or IP address
  - Reduces size of header (22 bytes required)
  - Enhances privacy
- WSMP encompasses Transport Layer (TCP, UDP)

CRL Distribution in VANETs



1. CA generates CRL
2. CRL divided into N pieces using coding method
3. Coded pieces distributed to RSUs
4. V2I and V2V used to distribute CRL to all vehicles

V2I/V2V Distribution



1. OBU in range of RSU
2. OBU uses V2I to download file pieces
3. OBU in range of other OBU
4. OBUs use V2V to download file pieces
5. Pieces downloaded

Application Layer

- Modified “OnOffApplication” and “PacketSink” to send and receive CRL pieces.
- Add CRL piece information to packet
- Use QosTag to set user priority of packets
- Use PeekData() in sink to find CRL piece information
- Keep track of Piece IDs and piece count with bitset

Application Layer

- OnOffApplication parameters:

```
Config::SetDefault ("ns3::OnOffApplication::OnTime",
  RandomVariableValue (ConstantVariable (syncInterval/2-
  guardInterval))); // application active
Config::SetDefault ("ns3::OnOffApplication::OffTime",
  RandomVariableValue (ConstantVariable
  (syncInterval/2+guardInterval))); // application paused
```

- Install Control Channel application:

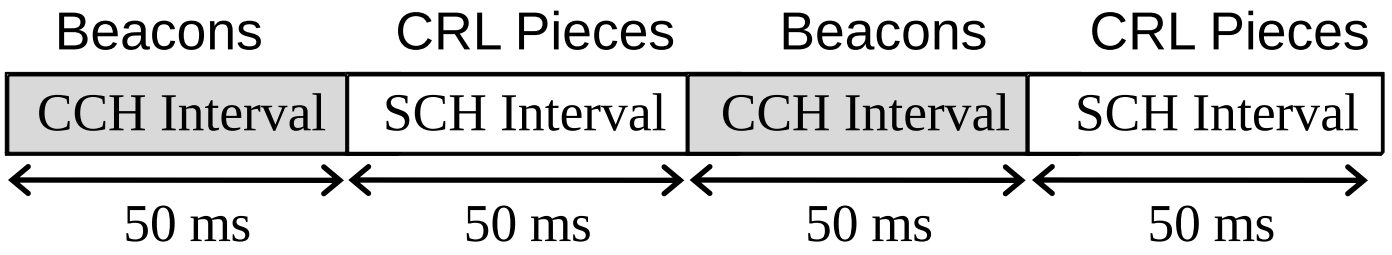
```
Ptr<Node> appRSU = nc_RSU.Get(i);
Ptr<OnOffApplication> cchRSU = CreateObject<OnOffApplication>();
   appRSU->AddApplication(cchRSU);
   cchRSU->SetPacketSize(beaconSize);
   cchRSU->SetMaxBytes(beaconSize);// only send one beacon per
  control channel interval
```

- Install Service Channel application:

```
Ptr<OnOffApplication> schRSU = CreateObject<OnOffApplication>();
   appRSU->AddApplication(schRSU);
   schRSU->SetPacketSize(packetSize);
   schRSU->SetMaxBytes(0); // continue sending data packets
```

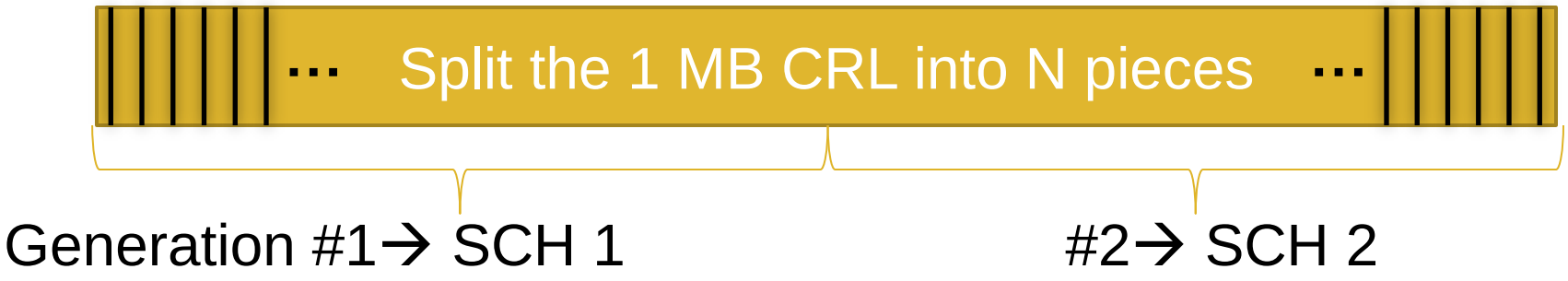
Most Pieces Broadcast

- Reduces the channel contention by limiting the number of nodes (OBUs, RSUs) that broadcast in a radio “footprint”
- Adds information to the control channel (CCH) beacon to advertise the count of CRL pieces
- Only the node with the most pieces broadcasts during SCH Interval

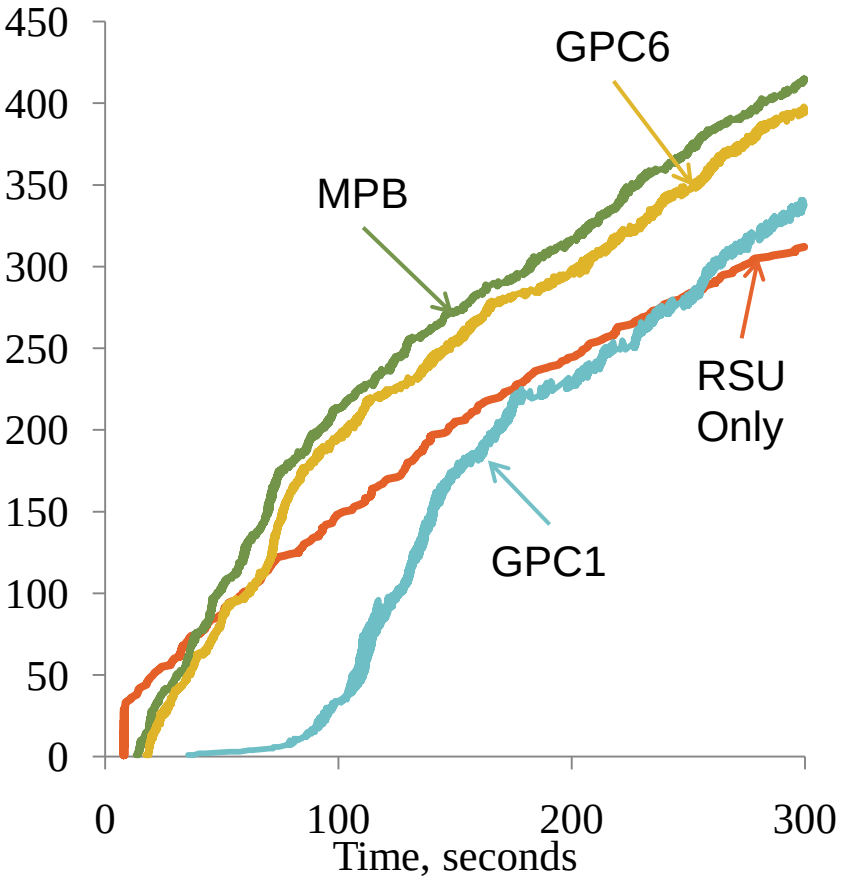


Generation per Channel

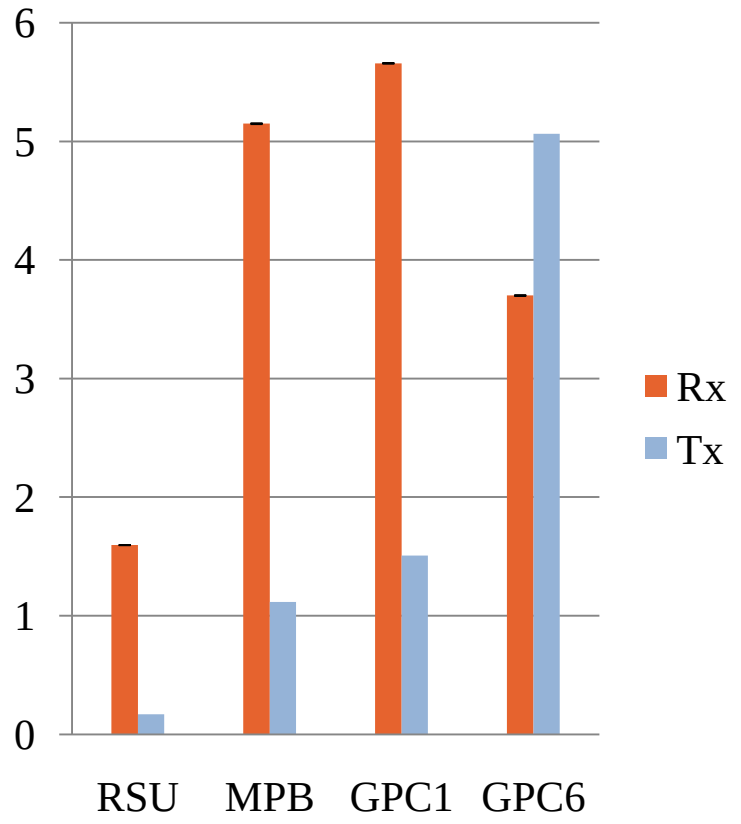
- Takes advantage of multiple service channels
- Using Network Coding, different file generations are distributed on different service channels
- Nodes visit different service channels to get the entire file
- All nodes contend for channel during SCH Interval



Results for 520 OBUs



Number of OBUs completing CRL download



Number of pieces during CRL download, in millions

Mobility Traces

- Traces from Naumov, Baumann, and Gross (2006), at ETH Zurich, Switzerland.
- Developed from Multi-agent Microscopic Traffic Simulator.
- Original trace file for 259,978 vehicles in a 354 kilometer by 263 kilometer area for over 21 hours.
- Traces divided into 300 second intervals for different settings (highway, city, mixed) and densities (low, medium, high).

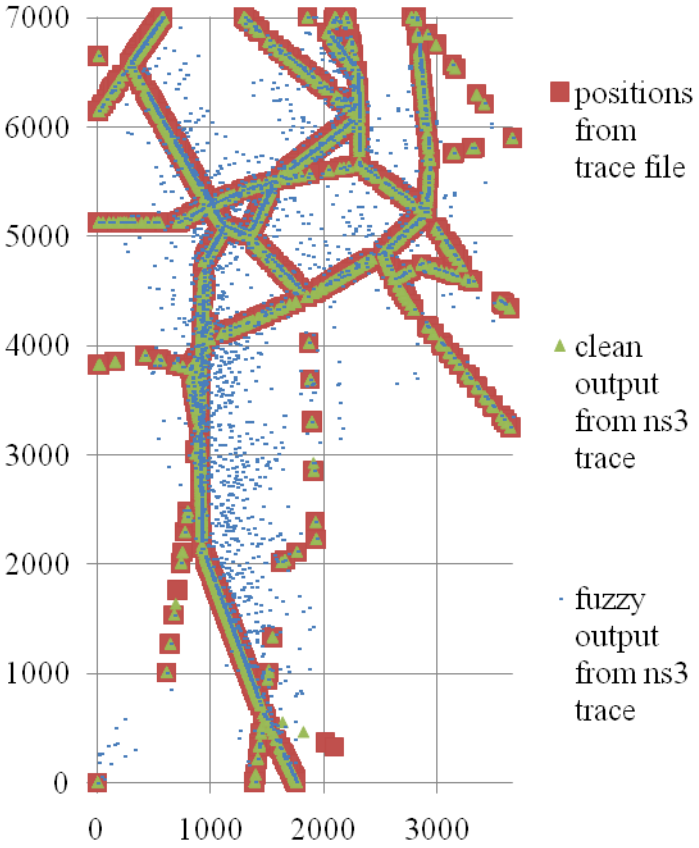
Mobility Traces

- Format of traces is:
  - x-destination, y-destination, velocity.
- Format of ns-3 reader is:
  - x-velocity, y-velocity, z-velocity.
- Adapted reader from Francesco Malandrino
  - Accept “switch ON” and “switch OFF” commands.

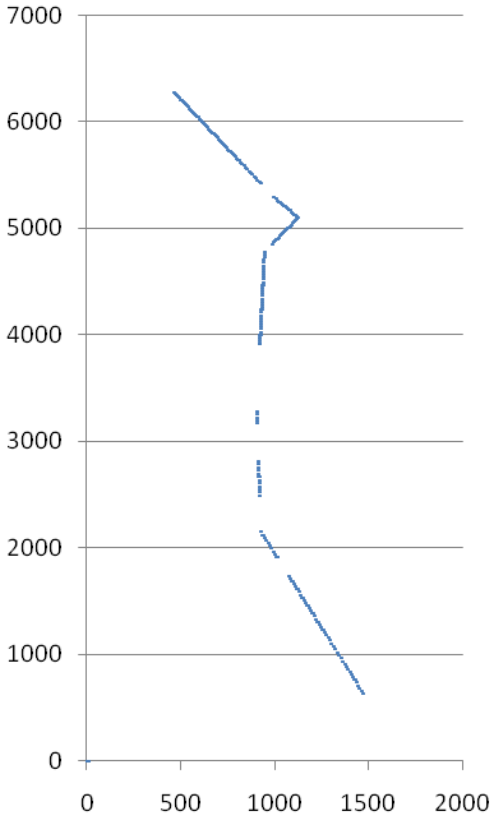
```
Simulator::Schedule (Seconds(pr->dvals[2]),&Ipv4::SetUp,
 (NodeList::GetNode(pr->ivals[3]+m_nRSU))->GetObject<Ipv4> (),1);
```

  - Correct rounding/timing from trace file.

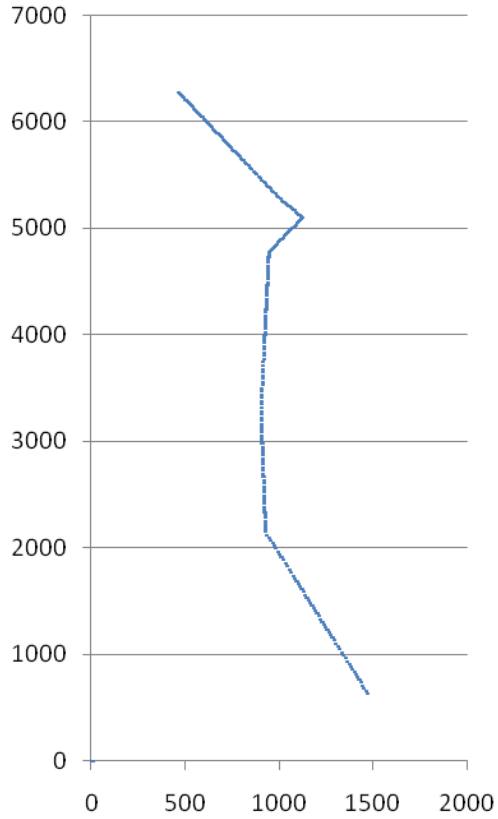
Output from Updated Trace Reader



Position trace of 520 OBUs using "CourseChangeCallback"



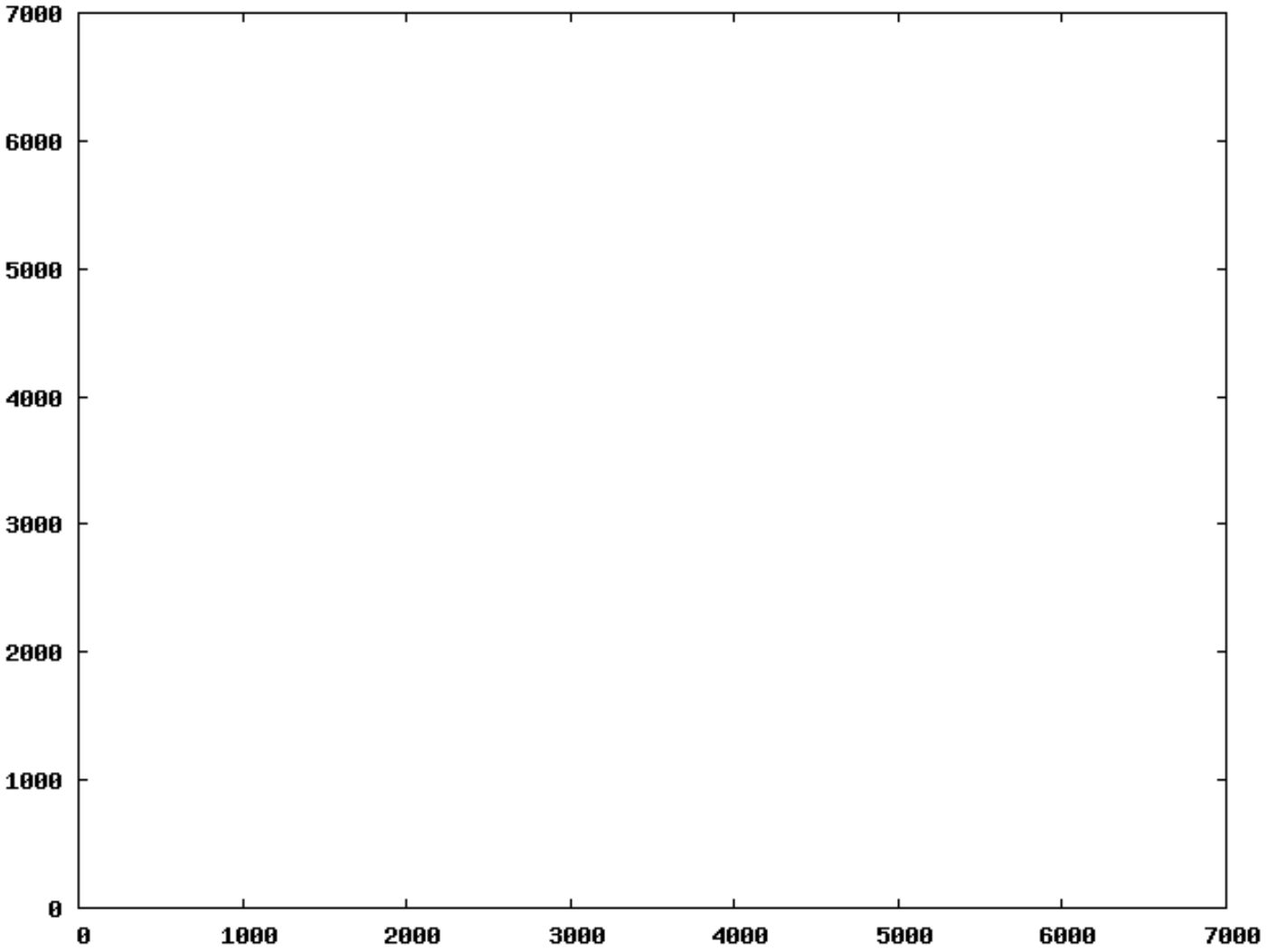
Original reader



Updated reader

Position trace of single OBU at 1 second intervals

Enge-Oberstrass



Questions?

CRL Distribution in VANETs using ns-3

Michael E. Nowatkowski

Workshop on ns-3

Malaga, Spain

15 March 2010

nsa Network Security and Architecture Laboratory

Georgia Tech College of Engineering
School of Electrical and Computer Engineering