Writing an OSTINATO PROTOCOL BUILDER

How to add more protocols to the Ostinato traffic generator

Srivats P.
(Creator, Ostinato)

What is Ostinato?

Packet Crafter ‘n Traffic Generator

Software only, Open Source

Cross Platform – Windows, MacOS, Linux

Ostinato – a brief history

- Started 2007
- Public launch Apr 2010 – v0.1
- The BIG ONE-O Sep 2019 – v1.0
- Lead developer-maintainer
- Side project
- Project sustained via paid binaries

Ostinato – features

- Multiple streams (stream = sequence of packets)
- Per stream: rates, bursts, no. of packets, stats
- Real time Port Statistics
- Device Emulation (ARP, Ping)
- Common protocols (VLAN, IP, TCP, UDP, IGMP etc.)
  - Stateless only; no stateful support
- Set/Edit value for any field of any protocol
- Vary packet fields
- PCAP import, edit and replay
- … and many more!

Wireshark in Reverse

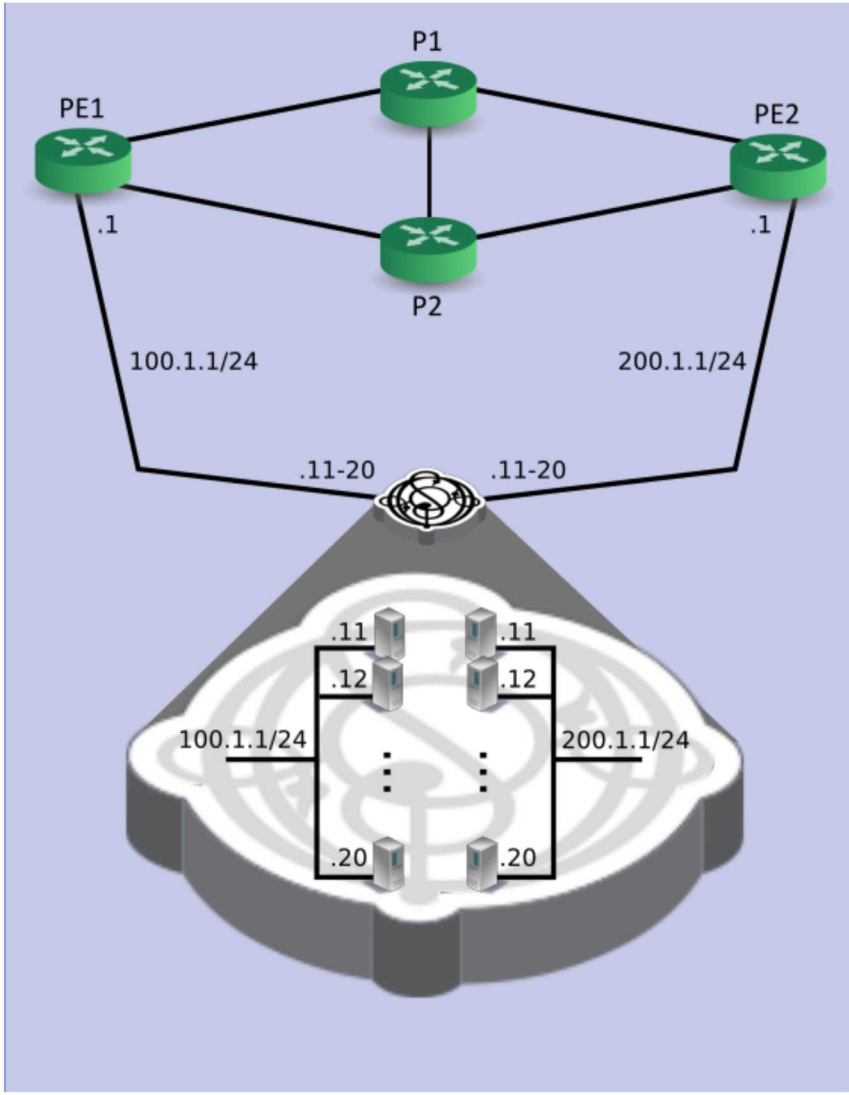
Demos and more

Learn more and
watch Ostinato demo videos

https://ostinato.org/docs/sharkfest2020

Ostinato demos

- Basics
  - Packets from scratch
  - Pcap import/edit/replay
- Scenarios
  - Routing test
  - IMIX test
  - IGMP/multicast test



Currently supported protocols

L2	L3	L4	L5	Misc
Mac	ARP	ICMP (v4/v6)	Text	HexDump
Eth2	IPv4	IGMP	Pattern Payload	UserScript
802.3	IPv6	MLD		
LLC	IPoverIP (v4/v6)	TCP		
SNAP	STP	UDP		
VLAN				

As of Jan 2021

New protocols can be added using Ostinato's
Protocol Builder Framework

Code - Language and dependencies

- Language
  - C++
  - Protocol buffers
- Dependencies
  - Qt 5.9+
  - Protobuf 2.6+
  - libpcap/npcap
  - libnl3, libnl-route3 (Linux only)

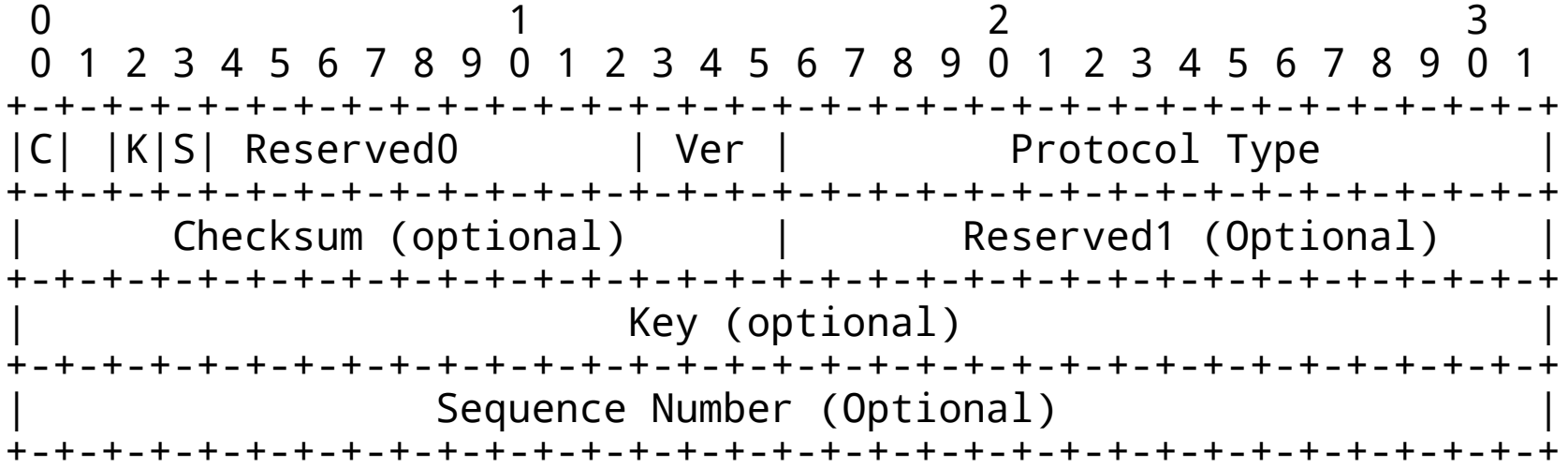
What is a Protocol Builder?

- Wireshark
  - Captures packets and dissects them
  - **“Protocol Dissectors”**
- Ostinato
  - Builds packets and puts them on the wire
  - **“Protocol Builders”**

Parts of a Protocol Builder

- Core
  - `sample.proto, {sample.pb.h, sample.pb.cc}, sample.h, sample.cpp`
- GUI widget
  - `sample.ui, {ui_sample.h}, sampleconfig.h, sampleconfig.cpp`
- PCAP import
  - `samplepdml.h, samplepdml.cpp`
- These are actual files from Ostinato code
  - template for new protocol builders
  - includes boilerplate code
  - and TODO instructions for protocol specific code

GRE – Frame Format



(RFC 2890)

GRE Protocol Builder files

- Copy and rename template files
  - `sample*.*` to `gre*.*`
  - Find & Replace `SAMPLE/Sample/sample` with `GRE/Gre/gre` in the gre*.* files
- New files
  - Core: `gre.proto, {gre.pb.h, gre.pb.cc}, gre.h, gre.cpp`
  - GUI widget: `gre.ui, {ui_gre.h}, greconfig.h, greconfig.cpp`
  - PCAP import: `grepdml.h, grepdml.cpp`
- Modified Files
  - Definition: `protocol.proto`
  - Registration: `protocolmanager.cpp, protocolwidgetfactory.cpp, pdmlreader.cpp`
  - Build/Project: `ostproto.pro, ostprotogui.pro`

GRE Core

`gre.proto, gre.h, gre.cpp`

GRE Core (gre.proto)

```
import "protocol.proto";

package OstProto;

// GRE Protocol
message Gre {
    optional uint32     flags = 1 [default = 0xa];
    optional uint32     rsvd0 = 2;
    optional uint32     version = 3;
    optional uint32     protocol_type = 4;
    optional uint32     checksum = 5;
    optional uint32     rsvd1 = 6;
    optional uint32     key = 7 [default = 0x2020bad7];
    optional uint32     sequence_num = 8;
}

extend Protocol {
    optional Gre gre = 405;
}
```

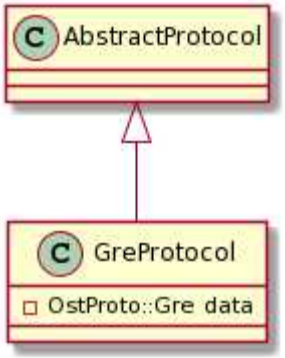
GRE Core (protocol.proto)

```
--- a/common/protocol.proto
+++ b/common/protocol.proto
@@ -156,6 +156,7 @@ message Protocol {
          kIcmpFieldNumber = 402;
          kIgmpFieldNumber = 403;
          kMldFieldNumber = 404;
+         kGreFieldNumber = 405;

          kTextProtocolFieldNumber = 500;
      }
```

GRE Core (gre.proto ==> gre.pb.[h,cc])

```
// protoc generated class

namespace OstProto {
// ...
class Gre : public ::google::protobuf::Message {
  // ...
  // optional uint32 flags = 1 [default = 10];
  inline bool has_flags() const;
  inline void clear_flags();
  static const int kFlagsFieldNumber = 1;
  inline ::google::protobuf::uint32 flags() const;
  inline void set_flags(::google::protobuf::uint32 value);
  // ...
}
}
```

GRE Core class (diagram)



GRE Core (gre.h)

```
#define GRE_FLAG_CKSUM 0x8
#define GRE_FLAG_KEY   0x2
#define GRE_FLAG_SEQ   0x1

class GreProtocol : public AbstractProtocol
{
public:
    enum grefield
    {
        // Frame Fields
        gre_flags = 0,
        gre_rsvd0,
        gre_version,
        gre_protocol,
        gre_checksum,
        gre_rsvd1,
        gre_key,
        gre_sequence,

        gre_fieldCount
    };

    // ...

private:
    OstProto::Gre data;
}
```

GRE Core (gre.cpp) – Boilerplate functions

```
// Same as template - no need to modify

AbstractProtocol* GreProtocol::createInstance(StreamBase *stream,
    AbstractProtocol *parent) {
    return new GreProtocol(stream, parent);
}

quint32 GreProtocol::protocolNumber() const {
    return OstProto::Protocol::kGreFieldNumber;
}

void GreProtocol::protoDataCopyInto(OstProto::Protocol &protocol) const {
    protocol.MutableExtension(OstProto::gre)->CopyFrom(data);
    protocol.mutable_protocol_id()->set_id(protocolNumber());
}

void GreProtocol::protoDataCopyFrom(const OstProto::Protocol &protocol) {
    if (protocol.protocol_id().id() == protocolNumber() &&
            protocol.HasExtension(OstProto::gre))
        data.MergeFrom(protocol.GetExtension(OstProto::gre));
}
```

GRE Core (gre.cpp) – Names and IDs

```
QString GreProtocol::name() const {
    return QString("General Routing Encapsulation Protocol");
}

QString GreProtocol::shortName() const {
    return QString("GRE");
}

AbstractProtocol::ProtocolIdType GreProtocol::protocolIdType() const {
    return ProtocolIdEth; // Types are ProtocolId[None|Llc|Eth|Ip|TcpUdp]
}

quint32 GreProtocol::protocolId(ProtocolIdType type) const {
    switch(type) {
        case ProtocolIdIp: return 47;
        default:break;
    }

    return AbstractProtocol::protocolId(type);
}
```

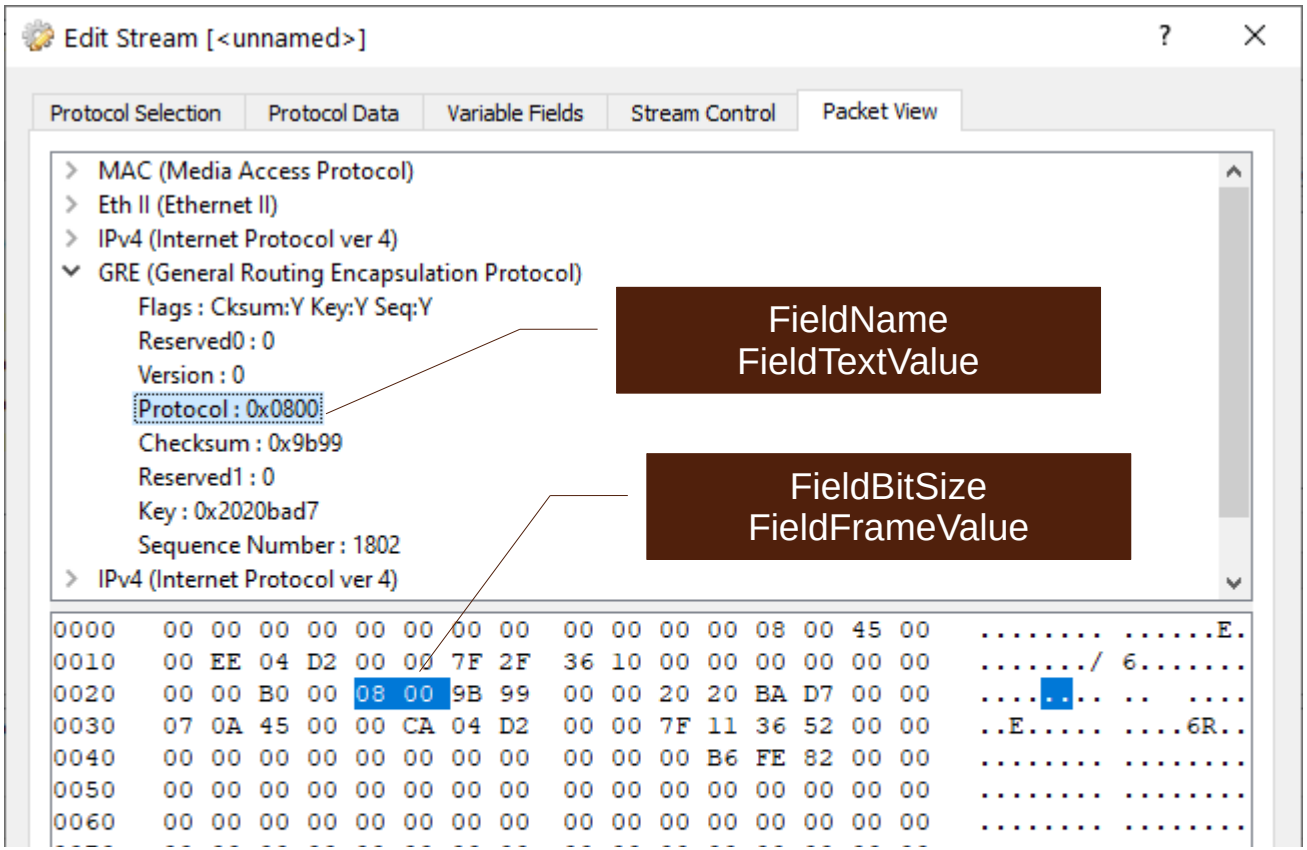
GRE Core (gre.cpp) – Field count and flags

```
int GreProtocol::fieldCount()  const {
    return gre_fieldCount;
}

AbstractProtocol::FieldFlags GreProtocol::fieldFlags(int index) const {
    AbstractProtocol::FieldFlags flags // Options: [Frame,Cksum,Meta]Field
                = AbstractProtocol::fieldFlags(index);// returns FrameField
    if (index == gre_checksum)
        flags |= CksumField;
    return flags;
}

int GreProtocol::frameFieldCount() const { // Only if no. of fields may vary
    int count = 4; // mandatory fields - flags, rsvd0, version, protocol
    if (data.flags() & GRE_FLAG_CKSUM)
        count += 2; // checksum, rsvd1
    if (data.flags() & GRE_FLAG_KEY)
        count++;
    if (data.flags() & GRE_FLAG_SEQ)
        count++;
    return count;
}
```

GRE Core (gre.cpp) – Field Data

```
QVariant GreProtocol::fieldData(int index, FieldAttrib attrib, int streamIndex) const {
    switch (index) {
        case gre_protocol: {
            quint16 protocol = payloadProtocolId(ProtocolIdEth);
            switch(attrib) {
                case FieldName: // as string
                    return QString("Protocol");
                case FieldValue: // as integer
                    return protocol;
                case FieldFrameValue: { // as a byte array in Network byte order
                    QByteArray fv;
                    fv.resize(2);
                    qToBigEndian(protocol, (uchar*) fv.data());
                    return fv;
                }
                case FieldTextValue: // value as string
                    return  QString("0x%1").arg(protocol, 4, BASE_HEX, QChar('0'));
                case FieldBitSize: // required ONLY if size is not a byte multiple
                    return 16;
                default:
                    break;
            }
            break;
        }
        // other fields ...
    }
    return AbstractProtocol::fieldData(index, attrib, streamIndex);
}
```

GRE Core (gre.cpp) – Field Data Attributes



GRE Core (gre.cpp) – Set Field Data

```
bool GreProtocol::setFieldData(int index, const QVariant &value,
        FieldAttrib attrib) {
    bool isOk = false;

    if (attrib != FieldValue)
        goto _exit;

    switch (index) {
        case gre_flags: {
            uint flags = value.toUInt(&isOk);
            if (isOk)
                data.set_flags(flags);
            break;
        }

        // ...
    }

_exit:
    return isOk;
}
```

GRE Core (gre.cpp) – Protocol Frame Size

```
//  Implement only if not a fixed size
int GreProtocol::protocolFrameSize(int /*streamIndex*/) const {
    int size = 4; // mandatory fields - flags, rsvd0, version, protocol

    if (data.flags() & GRE_FLAG_CKSUM)
        size += 4;
    if (data.flags() & GRE_FLAG_KEY)
        size += 4;
    if (data.flags() & GRE_FLAG_SEQ)
        size += 4;

    return size;
}
```

GRE Core - Registration

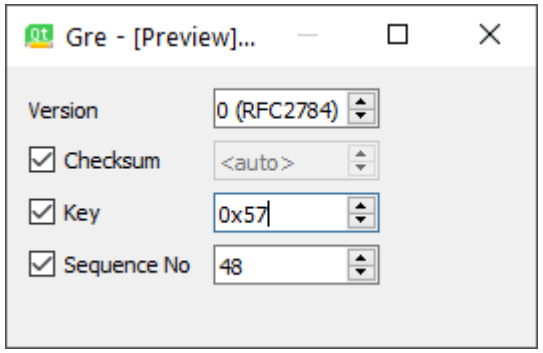
```
--- a/common/protocolmanager.cpp
+++ b/common/protocolmanager.cpp
@@ -46,6 +46,7 @@ along with this program.  If not, see <http://www.gnu.org/licenses/>
 #include "ip6over6.h"

 // L4 Protos
+#include "gre.h"
 #include "icmp.h"
 #include "igmp.h"
 #include "mld.h"
@@ -112,6 +113,8 @@ ProtocolManager::ProtocolManager()
             (void*) Ip6over6Protocol::createInstance);

     // Layer 4 Protocols
+    registerProtocol(OstProto::Protocol::kGreFieldNumber,
+            (void*) GreProtocol::createInstance);
     registerProtocol(OstProto::Protocol::kIcmpFieldNumber,
             (void*) IcmpProtocol::createInstance);
     registerProtocol(OstProto::Protocol::kIgmpFieldNumber,
```

```
--- a/common/protocol.proto
+++ b/common/protocol.proto
@@ -156,6 +156,7 @@ message Protocol {
         kIcmpFieldNumber = 402;
         kIgmpFieldNumber = 403;
         kMldFieldNumber = 404;
+        kGreFieldNumber = 405;

         kTextProtocolFieldNumber = 500;
     }
```

GRE GUI

`gre.ui, greconfig.h, greconfig.cpp`

GRE GUI (gre.ui)

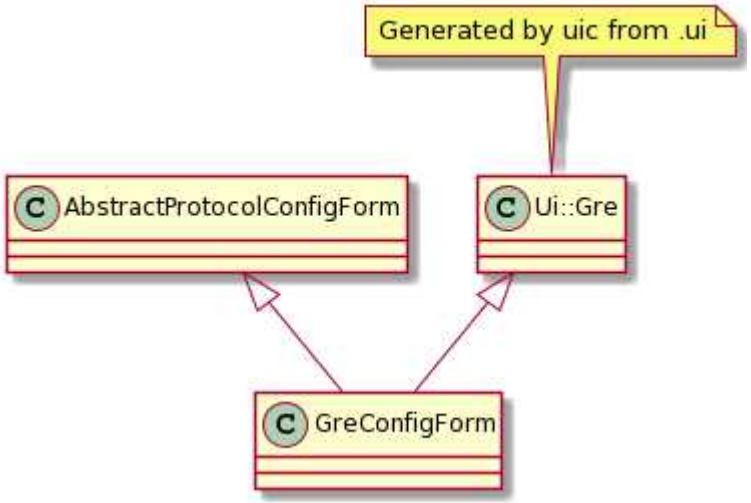


Created using Qt Designer
Saved as a .ui file (XML)

`gre.ui --> (uic) --> ui_gre.h`

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Gre</class>
 <widget class="QWidget" name="Gre">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>264</width>
    <height>140</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Gre</string>
  </property>
  <layout class="QGridLayout" name="gridLayout">
   <item row="0" column="0">
    <widget class="QLabel" name="label">
     <property name="text">
      <string>Version</string>
     </property>
    </widget>
   </item>
   <item row="0" column="1">
    <widget class="QSpinBox" name="version">
     <property name="specialValueText">
      <string>0 (RFC2784)</string>
     </property>
     <property name="maximum">
      <number>7</number>
     </property>
    </widget>
   </item>
   <!--- ... -->
 </widget>
</ui>
```

GRE GUI – Class Diagram



GRE GUI (greconfig.h)

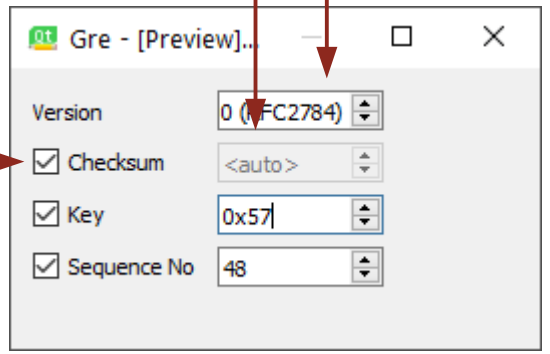
```
// Boilerplate from template

#include "abstractprotocolconfig.h"
#include "ui_gre.h"

class GreConfigForm :
    public AbstractProtocolConfigForm,
    private Ui::Gre
{
    Q_OBJECT
public:
    GreConfigForm(QWidget *parent = 0);
    virtual ~GreConfigForm();

    static GreConfigForm* createInstance();

    virtual void loadWidget(AbstractProtocol *proto);
    virtual void storeWidget(AbstractProtocol *proto);
};
```

GRE GUI (greconfig.cpp) – Boilerplate functions

```
GreConfigForm::GreConfigForm(QWidget *parent)
    : AbstractProtocolConfigForm(parent) {
    setupUi(this);
}

GreConfigForm::~GreConfigForm() {
}

GreConfigForm* GreConfigForm::createInstance() {
    return new GreConfigForm;
}
```

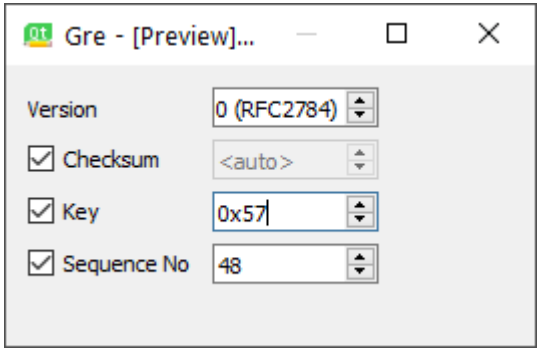
GRE GUI (greconfig.cpp) – Load widget

```
// Load widget contents from proto
void GreConfigForm::loadWidget(AbstractProtocol *proto) {
    version->setValue(
        proto->fieldData(
            GreProtocol::gre_version,
            AbstractProtocol::FieldValue
        ).toUInt());

    uint flags = proto->fieldData(GreProtocol::gre_flags,
                                  AbstractProtocol::FieldValue)
                      .toUInt();

    hasChecksum->setChecked(flags & GRE_FLAG_CKSUM);
    checksum->setValue(
        proto->fieldData(
            GreProtocol::gre_checksum,
            AbstractProtocol::FieldValue
        ).toUInt());
    // ...
}
```



GRE GUI (greconfig.cpp) – Store widget

```
// Store widget contents into proto
void GreConfigForm::storeWidget(AbstractProtocol *proto) {
    uint flags = 0;

    if (hasChecksum->isChecked())
        flags |= GRE_FLAG_CKSUM;
    if (hasKey->isChecked())
        flags |= GRE_FLAG_KEY;
    if (hasSequence->isChecked())
        flags |= GRE_FLAG_SEQ;

    proto->setFieldData(
        GreProtocol::gre_flags,
        flags);

    proto->setFieldData(
        GreProtocol::gre_version,
        version->value());

    proto->setFieldData(
        GreProtocol::gre_checksum,
        checksum->value());
    // ...
}
```



GRE GUI – Widget registration

```
--- a/common/protocolwidgetfactory.cpp
+++ b/common/protocolwidgetfactory.cpp
@@ -40,6 +40,7 @@ along with this program.  If not, see <http://www.gnu.org/licenses/>
 #include "ip6over4config.h"
 #include "ip6over6config.h"
 // L4 Protocol Widgets
+#include "greconfig.h"
 #include "icmpconfig.h"
 #include "igmpconfig.h"
 #include "mldconfig.h"
@@ -124,6 +125,9 @@ ProtocolWidgetFactory::ProtocolWidgetFactory()
             (void*) Ip6over6ConfigForm::createInstance);

     // Layer 4 Protocols
+    OstProtocolWidgetFactory->registerProtocolConfigWidget(
+            OstProto::Protocol::kGreFieldNumber,
+            (void*) GreConfigForm::createInstance);
     OstProtocolWidgetFactory->registerProtocolConfigWidget(
             OstProto::Protocol::kIcmpFieldNumber,
             (void*) IcmpConfigForm::createInstance);
```

GRE Pcap Import

`grepdml.h, grepdml.cpp`

Ostinato Intelligent PCAP Import

- Uses tshark to convert PCAP to PDML
- PDML => Packet Data Markup Language (XML)
- PDML content is essentially Wireshark protocol dissector data
  - PDML is NOT standard – may change across Wireshark/tshark versions if protocol dissector changes
- Import relies essentially on PDML field name, size, value

```
<proto name="gre" showname="Generic Routing Encapsulation (IP)" size="16" pos="34">
  <field name="gre.flags_and_version" showname="Flags and Version: 0xb000" size="2" pos="34" show="0x0000b000"
value="b000">
    <field name="gre.flags.checksum" showname="1... .... .... .... = Checksum Bit: Yes" size="2" pos="34" show="1" value="1"
unmaskedvalue="b000"/>
    <field name="gre.flags.routing" showname=".0.. .... .... .... = Routing Bit: No" size="2" pos="34" show="0" value="0"
unmaskedvalue="b000"/>
    <field name="gre.flags.key" showname="..1. .... .... .... = Key Bit: Yes" size="2" pos="34" show="1" value="1"
unmaskedvalue="b000"/>
    <field name="gre.flags.sequence_number" showname="...1 .... .... .... = Sequence Number Bit: Yes" size="2" pos="34"
show="1" value="1" unmaskedvalue="b000"/>
```

GRE Pcap Import (grepdml.h)

```
// Boilerplate from template – add more, if required

#include "pdmlprotocol.h"

class PdmlGreProtocol : public PdmlProtocol
{
public:
    virtual ~PdmlGreProtocol();

    static PdmlProtocol* createInstance();

    virtual void preProtocolHandler(QString name,
            const QXmlStreamAttributes &attributes, int expectedPos,
            OstProto::Protocol *pbProto, OstProto::Stream *stream);
    virtual void prematureEndHandler(int pos, OstProto::Protocol *pbProto,
            OstProto::Stream *stream);
    virtual void postProtocolHandler(OstProto::Protocol *pbProto,
            OstProto::Stream *stream);

    void fieldHandler(QString name, const QXmlStreamAttributes &attributes,
            OstProto::Protocol *pbProto, OstProto::Stream *stream);
    virtual void unknownFieldHandler(QString name, int pos, int size,
            const QXmlStreamAttributes &attributes,
            OstProto::Protocol *pbProto, OstProto::Stream *stream);

protected:
    PdmlGreProtocol();
};
```

GRE Pcap Import (grepdml.cpp) – 1:1 fields

```
PdmlGreProtocol::PdmlGreProtocol()
{
    ostProtoId_ = OstProto::Protocol::kGreFieldNumber;

    // Wireshark's gre.flags_and_version is NOT 1:1 with Ostinato fields

    fieldMap_.insert("gre.proto", OstProto::Gre::kProtocolTypeFieldNumber);
    fieldMap_.insert("gre.checksum", OstProto::Gre::kChecksumFieldNumber);
    fieldMap_.insert("gre.offset", OstProto::Gre::kRsvd1FieldNumber);
    fieldMap_.insert("gre.key", OstProto::Gre::kKeyFieldNumber);
    fieldMap_.insert("gre.sequence_number",
                     OstProto::Gre::kSequenceNumFieldNumber);
}
```

GRE Pcap Import (grepdml.cpp) – unknown fields

```
void PdmlGreProtocol::unknownFieldHandler(QString name,
        int pos, int size, const QXmlStreamAttributes& attributes,
        OstProto::Protocol* proto, OstProto::Stream* stream)
{
    // <field name="gre.flags_and_version"
    //        showname="Flags and Version: 0xb000" size="2" pos="34"
    //        show="0x0000b000" value="b000">
    if (name == "gre.flags_and_version") {
        bool isOk;
        quint16 flagsAndVersion = attributes.value("value")
                                            .toUInt(&isOk, kBaseHex);
        OstProto::Gre *gre = proto
                                ->MutableExtension(OstProto::gre);

        gre->set_flags(flagsAndVersion >> 12);
        gre->set_rsvd0((flagsAndVersion & 0x0FFF) >> 3);
        gre->set_version(flagsAndVersion & 0x0007);
    }
}
```

GRE Pcap Import (grepdml.cpp) – Special Handling

```
// Implement if any special handling is required

void PdmlGreProtocol::preProtocolHandler(QString name,
        const QXmlStreamAttributes& attributes,
        int expectedPos, OstProto::Protocol* pbProto,
        OstProto::Stream* stream) {
    return;
}

void PdmlGreProtocol::postProtocolHandler(OstProto::Protocol* pbProto,
        OstProto::Stream* stream) {
    return;
}

void PdmlGreProtocol::prematureEndHandler(int pos,
        OstProto::Protocol* pbProto, OstProto::Stream* stream) {
    return;
}
```

GRE Pcap Import – Registration

```
--- a/common/pdmlreader.cpp
+++ b/common/pdmlreader.cpp
@@ -28,6 +28,7 @@ along with this program.  If not, see <http://www.gnu.org/
licenses/>

 #include "arppdml.h"
 #include "eth2pdml.h"
+#include "grepdml.h"
 #include "llcpdml.h"
 #include "icmppdml.h"
 #include "icmp6pdml.h"
@@ -59,6 +60,7 @@ PdmlReader::PdmlReader(OstProto::StreamConfigList
*streams)

     factory_.insert("arp", PdmlArpProtocol::createInstance);
     factory_.insert("eth", PdmlEthProtocol::createInstance);
+    factory_.insert("gre", PdmlGreProtocol::createInstance);
     factory_.insert("http", PdmlTextProtocol::createInstance);
     factory_.insert("icmp", PdmlIcmpProtocol::createInstance);
     factory_.insert("icmpv6", PdmlIcmp6Protocol::createInstance);
```

More Info

- GitHub Repo
  https://github.com/pstavirs/ostinato
- Full code of GRE Protocol Builder - see GitHub PR
  https://github.com/pstavirs/ostinato/pull/336
- Protocol Builder Documentation
  https://devguide.ostinato.org/ProtocolBuilderHOWTO.html

- Slides and Video of this talk
  https://ostinato.org/docs/fosdem2021

Thank You!

Questions?