# 1 Общая структура

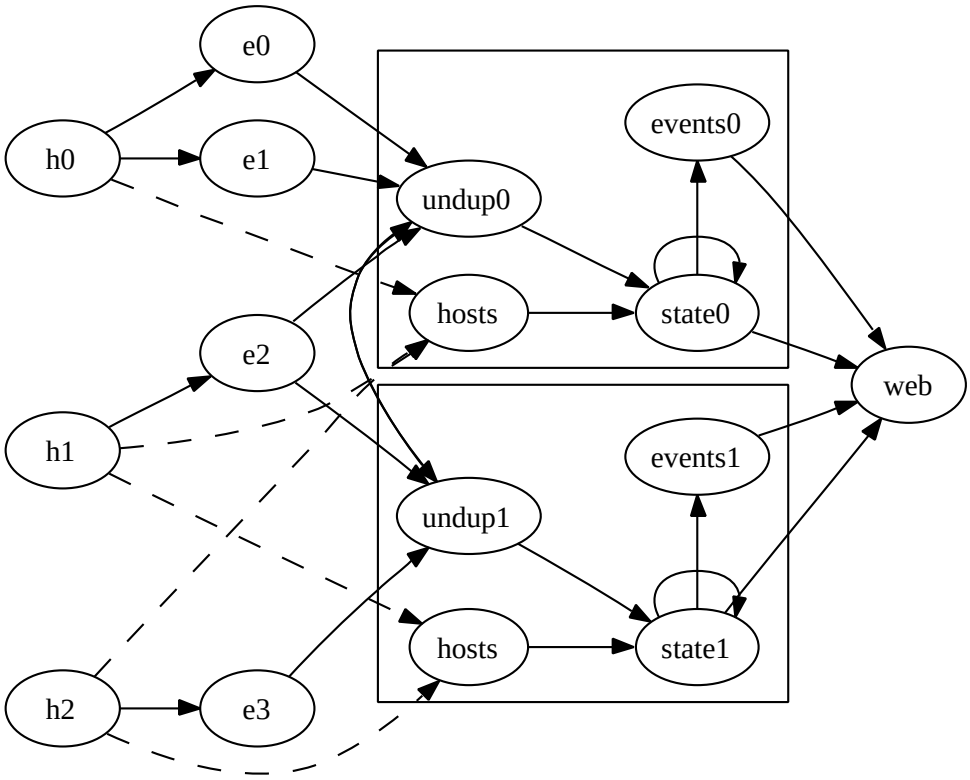


Рис. 1: Структура сети.

# 2 Элементы

Структура связей элементов.

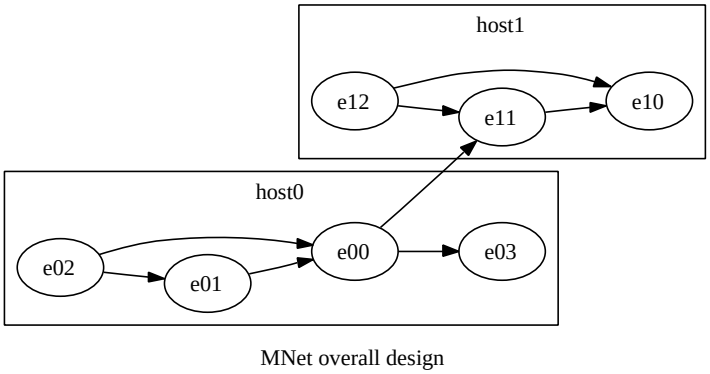


Рис. 2: Структура связей.

# 3 Безопасность

## 3.1 Модель доверия

Каждый хост, генерирующий сообщения имеет ключ, которым подписывает их. При этом, сообщение относящееся не к данному хосту, но подписанное им считается некорректным. Проверка осуществляется на входе в систему обработки. После этого подпись хоста несет лишь информационный характер и может быть удалена. Передача между отдельными подсистемами осуществляется при полном доверии между ними. Данная система достаточно проста в реализации и, в общем-то, достаточно надежна. Компрометация хоста может привести только к искажению информации им создаваемой. При этом компрометация промежуточного узла может привести к нарушению работы всяй системы. При этом могут существовать обработчики, которые лишь потребляют собощения.

# 4 Компоненты

Логически система разбивается на следующие компоненты

– Препроцессор конфигов. Возможно использование XSLT для упрощения конфигурационных файлов.

+ Генератор сообщений. Собственно код отвечающий за выполнение тестов, сбор информации и передачу ее шине.

+ Шина сообщений. Механизм распространения сообщений внутри системы.

– Синхронизация. Обеспечение устойчивости системы к потерям в некоторых каналах.

– Агрегация. Уменьшение потока сообщений за счет объединения схожих сообщений (например, при построении графика загрузки канала).

– Обработчик. Собственно обработка данных, принятие решения о той или иной реакции.

## 4.1 Генератор сообщений

Код обработки описания хоста и выполнения тестов. Есть примитивная реализация, написано три теста — `uname`, `uptime` и `ifflags`.

## 4.2 Шина сообщений

Возможно использование какой-нибудь левой. Стоит посмотреть на разные.

- dbus Работает только внутри одного хоста.
- spread Нетривиальный демон, непонятна система авторизации.
- mbus Несложный протокол обмена сообщениями. Есть RFC. Есть питоновская морда. Использует multicast/unicast сообщения.

# 5 Реализация

## 5.1 Языки

Ниже перечислены возможные языки, их плюсы, минусы и возможная роль. Использование C/C++ пока не предполагается.

### 5.1.1 XML

+ распространенность, наличие парсеров практически во всех языках
+ human readable; значительно облегчает отладку и добавление новых
+ древовидная расширяемая структура
− небольшое увеличение объема

Роль: формат сообщений и хранимое представление данных, конфигурационные файлы

### 5.1.2 XSLT

+ наиболее удобный способ переводить XML в HTML
+ простой способ манипуляции над XML
+ псевдо-функциональность, простое встраивание в скрипты
− сложность написания объемных алгоритмов
− нетривиальные глюки реализаций

Роль: преобразование на этапе представления пользователю, компиляция конфигурационных файлов, несложные манипуляции над состояниями

### 5.1.3 Shell Script Suite (bash, awk, sed, ...)

+ распространенность, почти любая *NIX система
+ скорость и простота написания конечных скриптов
+ использование программ написаных на любых языках
− сложность написания statefull программ

Роль: скрипты вытягивающие информацию из системы, склейки

### 5.1.4 Python

+ скриптовый, но с возможностью компиляции
+ простота использования в качестве языка-склейки различных частей
− глючные движки
− неявная типизация, облегчающие появление ошибок

Роль: склейка внутренних компонент, возможно — реализация отдельных частей

### 5.1.5 Haskell

+ удобство описания алгоритмов
+ возможность прямой компиляции в документацию
− сложная самантика
− слишком высокоуровневый

Роль: язык общего описания структуры, возможно — реализация компилятора состояний

## 5.2 Транспортный протокол

Для пассивных коммуникаций имеет смысл использовать в качестве транспортного уровня http (GET). Даже если внутри хоста используется какой-то другой протокол, довольно просто пишется web-гейт в него.

Для активного взаимодействия возможно использование либо http (POST), либо написание простого сетевого сервера работающего с XML сообщениями фиксированного формата. В последнем случае возможно использование клиента на базе программ netcat или socket.

# 6 Приложение

## 6.1 Пример описания хоста

```
<?xml version='1.0' encoding='utf-8'?>
<?xml-stylesheet type='text/xsl' href='mnet-host.xslt'?>
<mnet:hostmap xmlns:mnet='http://psha.org.ru/ns/mnet'>
<mnet:host uuid='void.local'>
        <mnet:detail>Центральный фтп</mnet:detail>
        <mnet:element type='service' uuid='void.local/routing' class='routing'>
                <mnet:detail>Network routing</mnet:detail>
                <mnet:software>iproute2</mnet:software>
                <mnet:software>iptables</mnet:software>
        </mnet:element>
        <mnet:element type='service' uuid='void.local/dns' class='dns'>
                <mnet:detail>Secondary DNS; Slave at bi.local</mnet:detail>
                <mnet:software>bind9</mnet:software>
                <mnet:test name='host'>
                        <mnet:param name='dns'>127.0.0.1</mnet:param>
                        <mnet:param name='host'>www.ru</mnet:param>
                </mnet:test>
                <mnet:test name='host' host='bi.local'>
                        <mnet:param name='dns'>void.local</mnet:param>
                        <mnet:param name='host'>www.ru</mnet:param>
                </mnet:test>
                <mnet:test name='host'>
                        <mnet:param name='host'>local</mnet:param>
                        <mnet:param name='mode'>serial</mnet:param>
                </mnet:test>
        </mnet:element>
        <mnet:element type='service' uuid='void.local/ftp' class='ftp'>
                <mnet:detail>Internal FTP</mnet:detail>
                <mnet:software>vsftpd</mnet:software>
        </mnet:element>

        <mnet:element uuid='void.local/core' class='core'>
                <mnet:detail>void.local core</mnet:detail>
                <mnet:hardware>Athlon 1600+</mnet:hardware>
                <mnet:hardware>Asus MB</mnet:hardware>
                <mnet:software>debian/unstable</mnet:software>
                <mnet:test name='uname'/>
                <mnet:test name='uptime'/>
        </mnet:element>
        <mnet:element uuid='void.local/hda' class='disk'>
                <mnet:detail>40G Seagate disk</mnet:detail>
                <mnet:element ref='local' severity='critical'>core</mnet:element>
        </mnet:element>
        <mnet:element uuid='void.local/eth0' class='net'>
                <mnet:detail>e100</mnet:detail>
                <mnet:hardware>e100</mnet:hardware>
                <mnet:element ref='local' severity='critical'>core</mnet:element>
                <mnet:link severity='critical' uuid='void.local/link/internal'>
                        <mnet:ip tag='lz'>10.22.16.254</mnet:ip>
                </mnet:link>
        </mnet:element>
        <mnet:element uuid='void.local/eth1' class='net'>
                <mnet:detail>e100</mnet:detail>
                <mnet:hardware>e100</mnet:hardware>
                <mnet:element ref='local' severity='high'>core</mnet:element>
                <mnet:link severity='critical' uuid='void.local/link/external'>
                        <mnet:ip tag='m9'>10.35.40.2</mnet:ip>
                        <mnet:ip tag='real'>82.33.13.98</mnet:ip>
                </mnet:link>
        </mnet:element>
        <mnet:element uuid='void.local/tun2' class='net'>
                <mnet:detail>Tunnel to north segment</mnet:detail>
                <mnet:software>vtun</mnet:software>
                <mnet:element ref='local' severity='normal'>routing</mnet:element>
                <mnet:link severity='critical' uuid='void.local/link/north'></mnet:link>
        </mnet:element>
</mnet:host>
</mnet:hostmap>
```

# 7 Функциональная запись

## 7.1 Общее описание типов

```
data State = State
  deriving (Eq, Ord, Show, Read)
data ElRef = ElRef ElId Int
  deriving (Eq, Ord, Show, Read)
data Element = Element ElId State [ElRef]
  deriving (Eq, Ord, Show, Read)
data Host = Host HostId [Element]
  deriving (Eq, Ord, Show, Read)
data Message = Message{ msgId :: MsgId
  , msgChain :: ChainId
  , msgTs :: TimeStamp
  , msgBody :: String
  } deriving (Eq, Ord, Show, Read)
data MsgChain = MsgChain Message MsgChain | MsgTail ChainId
  deriving (Eq, Ord, Show, Read)
type MsgDB = [MsgChain]
type HostDB = [Host]
type MNetDB = (MsgDB, HostDB)
```

## 7.2 Вспомогательные

```
chainId :: MsgChain → ChainId
chainLookup :: MsgDB → ChainId → MsgChain
chainLookup db c = if cs ≡ [] then MsgTail c else (λ(x : ⊥) → x) cs where
  cs = filter (λx → chainId x ≡ c) db
```

## 7.3 Функторы

### 7.3.1 Предварительная обработка сообщений

```
chainUndup :: MsgChain → Message → (MsgChain, Maybe Message)
chainUndup _ :: MsgChain → Message → (MsgChain, Maybe Message)
chainUndup c m =
    if chainId c ≡ msgChain m then
```

```
      chainUndup_ c m
    else
      (c, Nothing)
  chainUndup_ (MsgTail u) m = (MsgChain m (MsgTail u), Just m)
  chainUndup_ (MsgChain m c) msg =
    if msgTs m < msgTs msg then
      (MsgChain msg (MsgChain m c), Just msg)
    else if msgId m ≡ msgId msg then
      (MsgChain m c, Nothing)
    else
      (MsgChain m c', msg')
    where
      (c', msg') = chainUndup c msg
  msgUndup :: MsgDB → Message → (MsgDB, Maybe Message)
  msgUndup db m = ((c : filter (((≢) (msgChain m)) ∘ chainId) db), m') where
    (c, m') = chainUndup (chainLookup db (msgChain m)) m
```

Функция добавляет в список сообщение в том и только том случае, если его еще там нет

### 7.3.2 Компиляция состояний

```
  stateCompile :: HostDB → Message → HostDB
```

Производит модификацию состояних хостов по полученым данным

### 7.3.3 Основной рабочий цикл

```
  mnetLoop :: MNetDB → [Message] → MNetDB
  mnetLoop d [] = d
  mnetLoop (m, h) (x : xs) = mnetLoop db' xs where
    db' = maybe (m, h) (λx'' → (m', stateCompile h x'')) x'
    (m', x') = msgUndup m x
```