



Data Format Description Language

and the

Apache Daffodil Incubator Project

Mike Beckerle, Tresys Technology
mbeckerle at tresys.com or at apache.org



Got Gnarly Data? - Come to BoF Session!

Wednesday 19:00 Terrasse

Use Apache Daffodil (Incubating) to parse your data into XML/JSON

- Bring an example of gnarly data
- We will create a schema for it
- Prize for gnarliest data format

Goal For Today

My goal in this talk is to...

Build Community for Apache Daffodil (Incubating)

- Recruit developers
- Find additional mentors
- Encourage integration of Daffodil into Apache projects that intake/export data

Convince you that....

- DFDL/Daffodil solves an important problem
- Daffodil is technology that would be *fun* to work on

Agenda



Motivation

- The Data Format Problem
- DFDL, standards, Daffodil, and why it is important
- DFDL Schemas

Technology



- DFDL (pronounced “DaFoDiL” or “Dee Eff Dee Ell”)
- Daffodil – the software
 - Code base details and status
 - Cool Stuff Daffodil/DFDL Does for You
 - Streaming unparsing with forward reference
- What's Next – Plans/Hopes/Dreams

Why is DFDL Needed?

There are *hundreds* of ad-hoc data format description systems

Every Enterprise Software Company

- IBM (10+)
- Oracle(10+)
- SAP(10+)
- Microsoft
- SAS
- SyncSort
- AbInitio
- Pervasive
- Qlik/Exprressor
- Dozens more

Every kind of software that takes in data:

- data directed routing
- database
- data analysis and/or data mining
- data cleansing
- master data management
- application integration

All these data format descriptions are:

- *proprietary*
- *ad-hoc*
- *incompatible*

Even within products of the same company!

Why DFDL is Needed?

Hundreds of data format description systems...
means:

- Investment is spread too thin
 - Tools for creating data formats are inadequate
 - No product is comprehensive enough
 - Difficulty is grossly underestimated
 - Some products aren't fast enough
- Customer lock in
- Inflexible packaging
 - Not libraries - must embed some product in your application data flow

Why DFDL is Needed - New Use Cases

Cybersecurity

- Normalization of data
 - Complete rip and rebuild
 - Break down data fully based on DFDL schema
 - Validate at fine granularity
 - Reassemble according to DFDL schema
- Removes a large class of data-borne threats
 - Data always is what it says it is!

Data Publishing

- Open Data mandates

Why DFDL is Needed - New Use Cases

Modernization

- Legacy data systems are still the source, *and target* for much processing
- Coexistence is required for successful incremental modernization

Skills Leverage

- Developers with XML/JSON skills are available
- Legacy data format skills are precious
 - Military data formats - Link16, VMF, USMTF, ...
 - COBOL and other FINSERV formats

Solving the Data Format Problem

An Open Standard DFDL

- Multiple implementations that interoperate
 - Commercial & Open Source 
- Long-term sponsors
 - IBM – has their own DFDL implementations
 - US DoD, Canada DND
 - Cybersecurity
- Available DFDL schemas for important data formats

A High-Quality Open Source Library Implementation

- With a supporting community of developers 
- With available commercial support (Tresys)

Data Format Description Language

DFDL: A new open standard

- From the Open Grid Forum (OGF)
- Work began in **2001**, accelerated around 2008
- Major contributors from UK, Canada, and USA
- First Implementation: IBM - November 2011
 - Business-oriented subset of DFDL language
- DFDL Specification - Version 1.0 – Sept. 2014
 - Thick - about 200 pages if you print it.
 - Allows "conforming subsets" - required core is small
- Proposed Recommendation - Status (as of Oct 2016)
 - Waiting for two-implementation interoperability demonstration
- DFDL Workgroup is active
 - Clarifications, Errata on v1.0

Data Format Description Language

DFDL is a way of *describing* data formats

- It is NOT a data format itself!

DFDL combines State-of-the-Art

- Union of capabilities across many marketplace data integration products/tools/packages

DFDL adds small number of real innovations

- Overcome limitations of prior-gen e.g.,
 - Computed Elements Capability
 - Expression language
 - BitOrder

Data Format Description Language

Core Concepts

- Leverage XML Schema (XSDL)
 - Grammar scaffolding
 - Describes the *logical* data model
 - DFDL uses only a *subset* of XML schema
 - Provides standard ways to annotate
- Add annotations
 - Describe the *physical* representation.
- Read and write from same DFDL Schema

Because Developers [Love | Hate] XML

- The DFDL Schema is based on XSDL
- The Infoset created when parsing data does NOT have to be XML

Example – Delimited Text Data

rlimit=5;rpngx=-7.1E8

Example – Delimited Text Data

rlimit=5;rpngx=-7.1E8

Initiator

Separator

Initiator

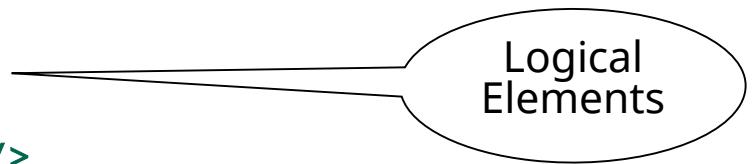
ASCII text
integer

ASCII text
floating point

Separators, initiators (aka tags), & terminators are all examples in DFDL of *delimiters*

DFDL Schema

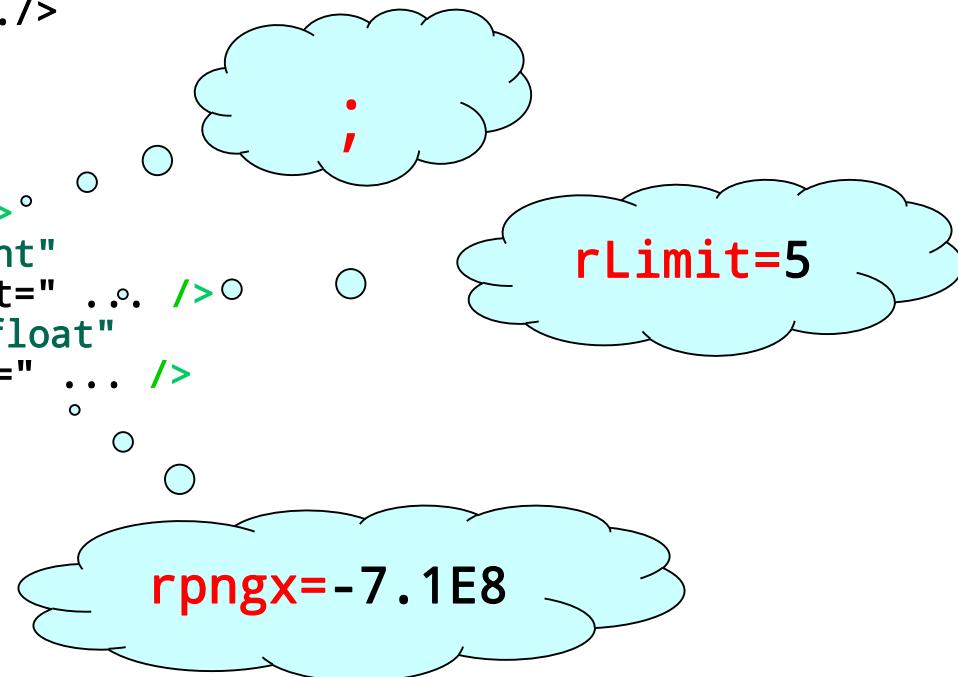
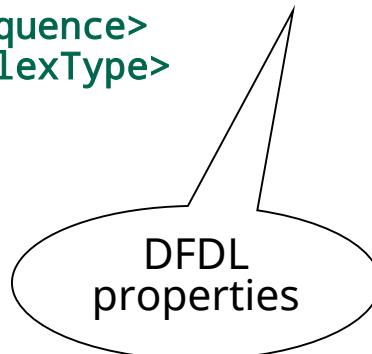
```
<xs:complexType name="rValues">
  <xs:sequence>
    <xs:element name="rlim" type="xs:int"/>
    <xs:element name="rpng" type="xs:float"/>
  </xs:sequence>
</xs:complexType>
```



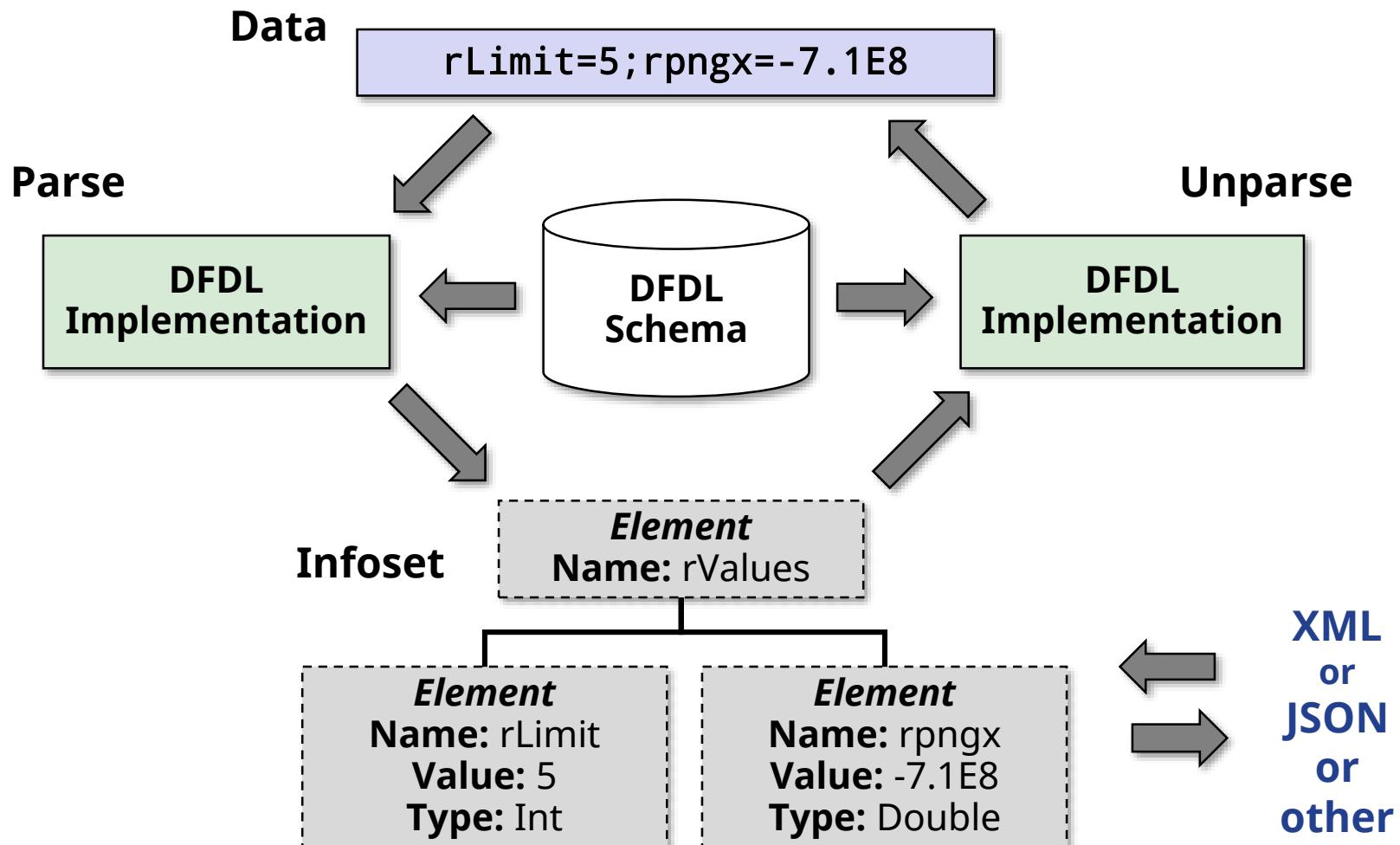
DFDL schema

```
<xs:annotation>
  <xs:appinfo source="http://www.ogf.org/dfdl/v1.0">
    <dfdl:format representation="text"
      textNumberRep="standard" encoding="ascii"
      lengthKind="delimited" .../>
  </xs:appinfo>
</xs:annotation>

<xs:complexType name="rValues">
  <xs:sequence dfdl:separator=";" ... >
    <xs:element name="rLim" type="xs:int"
      dfdl:initiator="rLimit=" ... />
    <xs:element name="rpng" type="xs:float"
      dfdl:initiator="rpngx=" ... />
  </xs:sequence>
</xs:complexType>
```



DFDL Data and InfoSet Lifecycle



More DFDL Properties

initiator	byteOrder	textCalendarJustification
terminator	bitOrder	calendarPatternKind
documentFinalTerminatorCanBeMissing	encoding	calendarCheckPolicy
output.NewLine	encodingErrorPolicy	calendarTimeZone
length	utf16Width	calendarObserveDST
lengthPattern	ignoreCase	calendarFirstDayOfWeek
textStringPadCharacter	alignment	calendarDaysInFirstWeek
textNumberPadCharacter	alignmentUnits	calendarCenturyStart
textCalendarPadCharacter	fillByte	binaryCalendarRep
textBooleanPadCharacter	leadingSkip	nilKind
escapeCharacter	trailingSkip	nilValueDelimiterPolicy
escapeBlockStart	lengthKind	useNilForDefault
escapeBlockEnd	lengthUnits	emptyValueDelimiterPolicy
escapeEscapeCharacter	prefixIncludesPrefixLength	sequenceKind
extraEscapedCharacters	prefixLengthType	hiddenGroupRef
textNumberPattern	representation	initiatedContent
textStandardGroupingSeparator	textPadKind	separatorPosition
textStandardDecimalSeparator	textTrimKind	separatorPolicy
textStandardExponentRep	textOutputMinLength	separatorSuppressionPolicy
textStandardInfinityRep	escapeKind	choiceLengthKind
textStandardNaNRep	generateEscapeBlock	choiceLength
textStandardZeroRep	textStringJustification	choiceDispatchKey
textBooleanTrueRep	textNumberRep	choiceBranchKey
textBooleanFalseRep	textNumberJustification	occursCountKind
calendarPattern	textNumberCheckPolicy	occursCount
calendarLanguage	textStandardBase	floating
binaryCalendarEpoch	textNumberRoundingMode	truncateSpecifiedLengthString
nilValue	textNumberRounding	decimalSigned
separator	textNumberRoundingIncrement	
occursStopValue	textZonedSignStyle	
inputValueCalc	binaryNumberRep	
outputValueCalc	binaryDecimalVirtualPoint	
textBidi	binaryNumberCheckPolicy	
textBidiTextOrdering	binaryPackedSignCodes	
textBidiOrientation	binaryFloatRep	
textBidiSymmetric		
textBidiTextShaped		
textBidiNumeralShapes		
	textBooleanJustification	
	binaryBooleanTrueRep	
	binaryBooleanFalseRep	

DFDL Schemas

* = in development
** = not yet published

Public (github)	MIL-STD-2045 PCAP NITF PNG JPEG NACHA vCard ShapeFile(.shp)	EDIFACT IBM4690-TLOG ISO8583 BMP GIF Praat TextGrid ARINC429* JPEG2000** planned: EP, DNG, WMF, EMF, ... planned: Asterix, IPFIX
FOUO (DI2E.net & Forge.mil)	VMF (MIL-STD-6017) USMTF ATO (MIL-STD-6040) LINK16 (NATO STANAG 5516/MIL-STD-6016) A-GNOSC REMEDY ARMY DRRS USCG UCOP CEF-R1965 GMTIF (STANAG 4607)	
Commercial License \$\$\$	SWIFT-MT (IBM) HIPAA-5010 (IBM) HL7-2.7 (IBM)	

Other DFDL Implementations

- IBM DFDL - The First DFDL Implementation - v1.0 Nov 2011
 - Embeddable as Library, C and Java
 - Includes Eclipse based tooling - graphical DFDL schema editor/test environment
 - Found in IBM products:
 - IBM InfoSphere Information Bus product family
 - IBM z/TPF product family
- European Space Agency - DFDL4S = DFDL for Space
 - Embeddable as Library, Java and C++ versions.
 - Binary-data-only subset of DFDL
 - Created by ESA for satellite data descriptions
 - Evolving to be a fully compatible DFDL subset.
 - More info at
 - <http://eop-cfi.esa.int/index.php/applications/dfdl4s>

Daffodil

The Open Source DFDL Implementation
aka Apache Daffodil (Incubating)

Daffodil - History

- Started out at University of Illinois/NCSA
 - Research project ~2009
 - ❤️ Written in **Scala** - runs on Java JVM
- Further developed by Tresys Technology ~2012
 - Funded by the US DoD, Canada DND
 - Open source from the start
 - Version 1.0 – parse only, XML – 2015-03
 - Version 2.0 – parse & unparse, XML + JSON – 2017-09
- Apache Incubator - started 2017-08
 - Version 2.1.0 – 2018-05
 - Version 2.2.0 - available now

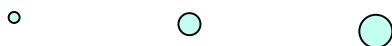
Avoid Version Confusion

*DFDL Language
Specification*

*Daffodil
Software*

- v1.0
-
- The diagram illustrates a mapping between two software components. On the left, under 'DFDL Language Specification', there is a single entry for 'v1.0'. An arrow points from this entry to a list of versions on the right, which are grouped by a curly brace. The list on the right, under 'Daffodil Software', includes 'v1.0.0', 'v1.1.0', 'v2.0.0', 'v2.1.0', 'v2.2.0', 'v3.x', and an ellipsis '....'.
- v1.0.0
 - v1.1.0
 - v2.0.0
 - v2.1.0
 - v2.2.0
 - v3.x
 -

Daffodil



If you download it,
what do you get?

- Jar libraries – runs on JVM
 - Compiler, runtime, utilities, TDML runner
- Command Line Interface
 - Interactive CLI debugger and trace
- Java & Scala API with documentation
- XML and JSON for parse-output, unparse-input

You must get your DFDL schemas somewhere...

- github (DFDLSchemas, others)
- Daffodil/DFDL project on DI2E.net/Forge.mil
- Write them! (and share them!)

Daffodil Internal Components

Compiler

- compiles DFDL schemas to runtime data structures
- Issues diagnostics

Scala API

Command Line Interpreter

Java API

Runtime

DPath - Xpath-like language

- compiler
- runtime

Parser primitives

InfoSet

- Convert to/from XML, JSON
- Fast, constant-time access

Unparser primitives

Breakpoint debugger

Utility Libraries

- for compiler
- for runtime

I/O library

- bits (not bytes), bitOrder
- streaming
- non-8-bit-characters (7, 6, 5)
- unbounded lookahead - parsing/backtracking, and unparsing

Tests - Unit
(Scala)

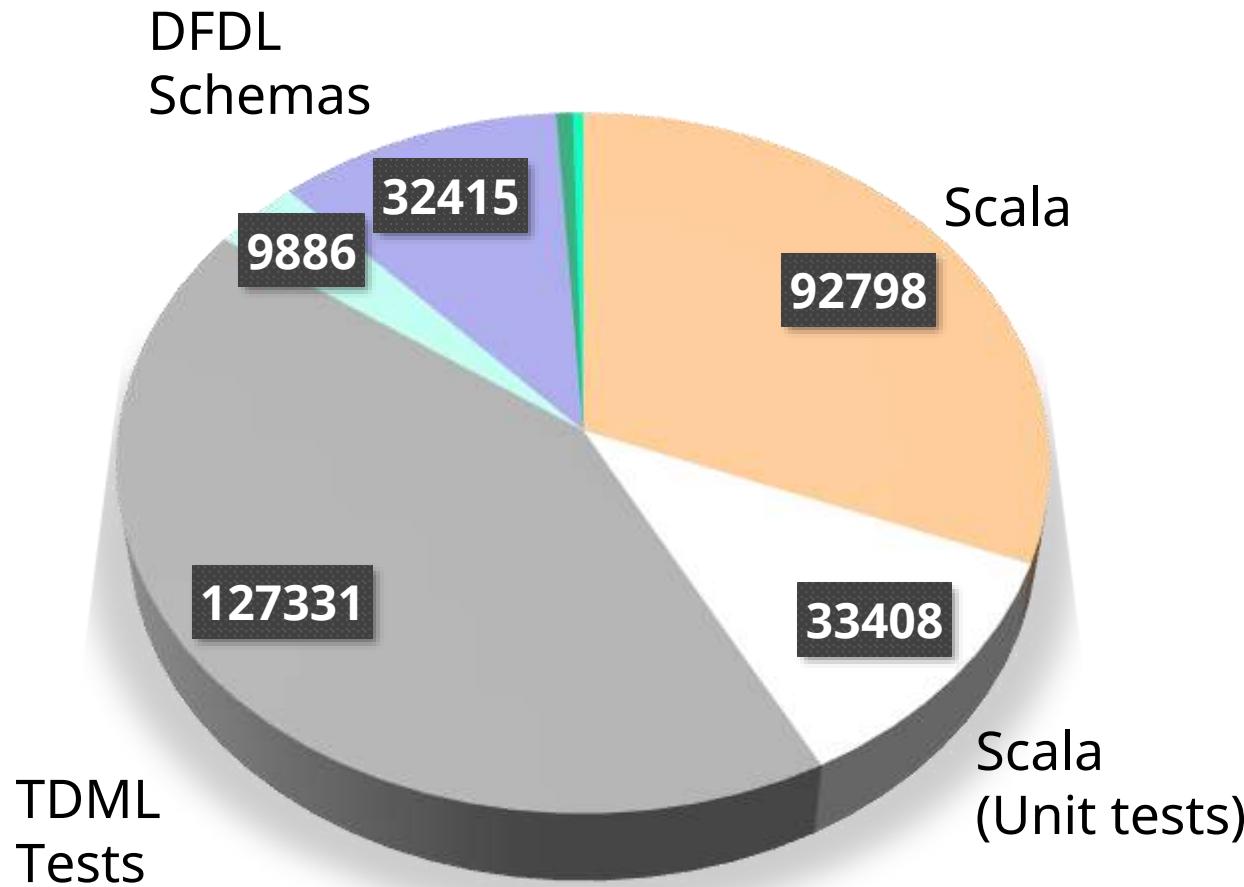
Tests - System
(TDML)

Written in Scala

TRESYs
Deep.

Daffodil Code Base

Lines = 298K Total



Data as of 2018-08-06

Daffodil Integrations

- Apache
 - Spark
 - NiFi
 - ...Your Project Here...
 - If you can intake/export XML or JSON, then Daffodil enables you to handle anything else describable with DFDL.
- Non-Apache
 - XProc - Calabash XML Pipeline Engine
 - Software AG webMethods™ Integration Server

Some Technical Coolness

Daffodil Runtime Implementation of
Streaming Unparser with Calculated-Value Elements

What is Harder: Parsing or Unparsing?

- Most CompSci people think parsing
 - backtracking - state saving/restoring
 - lookahead (bounded/unbounded)
- Unparsing seems simple in comparison
 - Just a tree-walk of the InfoSet outputting each element.....

But...

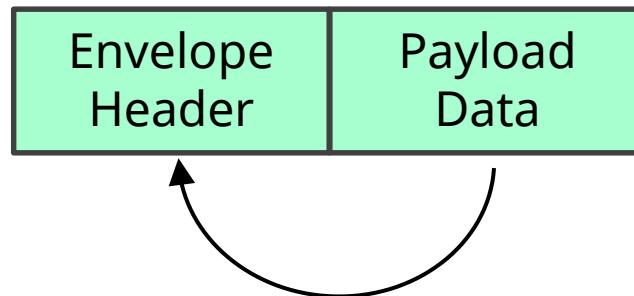
- DFDL has computed elements
 - New innovation over prior-gen format languages
- Let's take a look at what that means for unparsing

Streaming Unparsing

- Limit memory footprint
- Unparser consumes a stream of XML-like events (like SAX - Streaming API for XML)
- The Infoset tree does *NOT* exist
 - We did NOT just parse the data
 - We are starting from the XML events
- Infoset tree is built up as events arrive
 - Can be pruned when tree nodes are no longer needed
 - Infoset stays small even though data stream implies unbounded size

Common: Envelope + Payload

- Data Payload is surrounded by Envelope
- Envelope holds *Stored data length of payload*



length of payload data is stored in middle of envelope header

Stored Length Limitations on Streaming

```
<Packet>
  <PacketHeader>
    <Seconds>1371631556</Seconds>
    <USeconds>838904</USeconds>
    <InclLen> ??? </InclLen>
    <OrigLen> ??? </OrigLen>
  </PacketHeader>
  <pcap:LinkLayer>
    <pcap:Ethernet>
      <MACDest>005056E01449</MACDest>
      <MACSrc>000C29340BDE</MACSrc>
      <Ethertype>2048</Ethertype>
      . . .
    </pcap:Ethernet>
  </pcap:LinkLayer>
</Packet>
```

Unparse of Envelope needs length of Payload

Payload is a complex object. Must unparse it to determine its length.

Schema with Stored Length

```
<xs:element name="packetHeader">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Seconds" type="pcap:uint32"/>
      <xs:element name="USeconds" type="pcap:uint32"/>
      <xs:element name="InclLen" type="pcap:uint32" style="border: 2px solid blue; background-color: #f0f0ff; padding: 2px; margin-left: 10px;">
        ...
      />
    </xs:sequence>
  </xs:complexType>
</xs:element>
...
<xs:element ref="pcap:LinkLayer"
  dfdl:lengthUnits="bytes" dfdl:lengthKind="explicit"
  dfdl:length="{ ../header/InclLen }" style="border: 2px solid blue; background-color: #f0f0ff; padding: 2px; margin-left: 10px; margin-top: 10px; margin-bottom: 10px;"/>
```

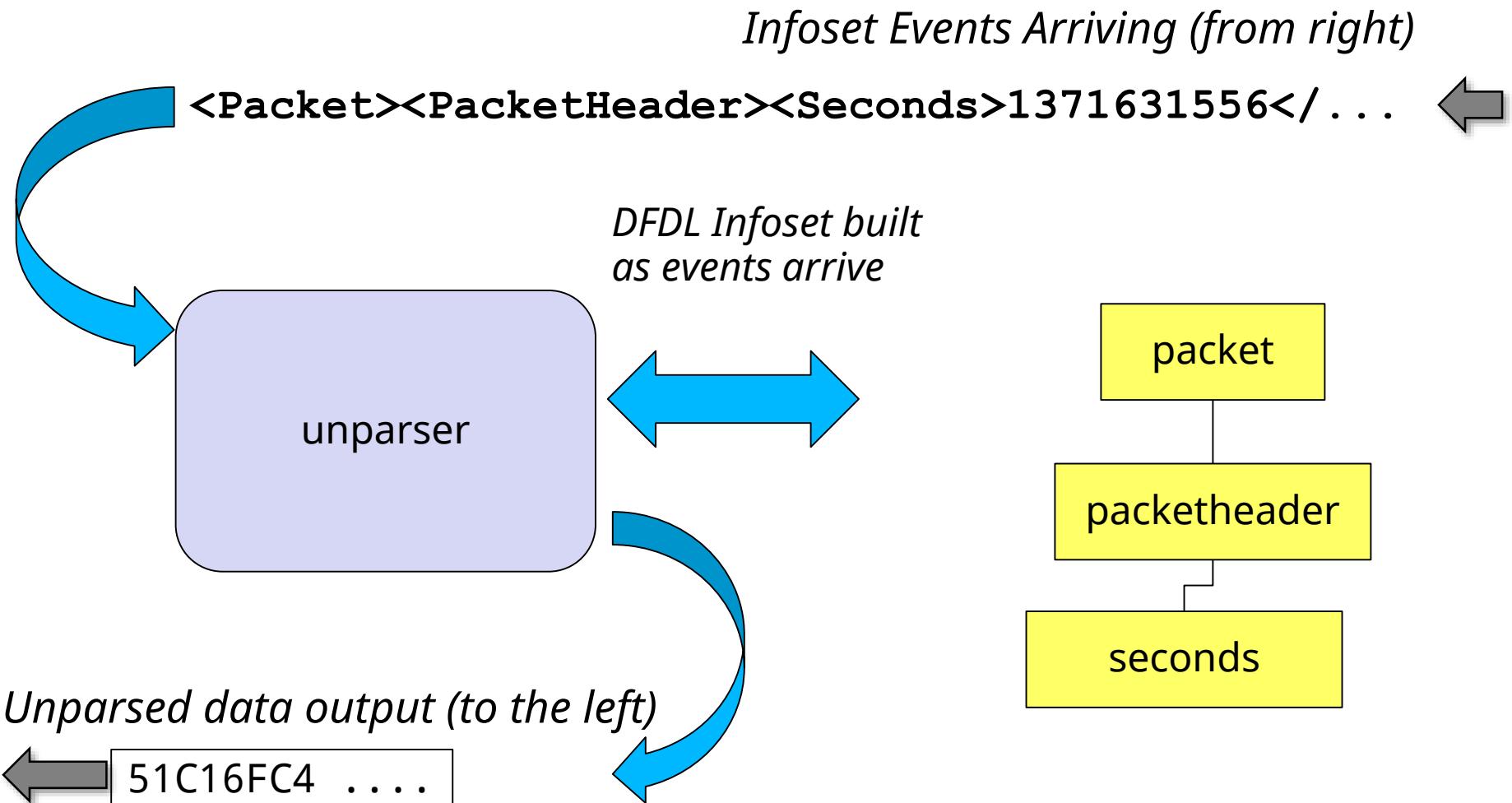
Schema with Stored Length

```
<xs:element name="packetHeader">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Seconds" type="pcap:uint32"/>
      <xs:element name="USeconds" type="pcap:uint32"/>
      <xs:element name="InclLen" type="pcap:uint32"
        dfdl:valueLength="{
          if (dfdl:valueLength(
            ../../pcap:LinkLayer/pcap:Ethernet,
            'bytes') le 60) then 60
          else
            dfdl:valueLength(
              ../../pcap:LinkLayer/pcap:Ethernet,
              'bytes') }"
        />
      ....
    </xs:sequence>
  </xs:complexType>
</xs:element>
...
<xs:element ref="pcap:LinkLayer"
  dfdl:lengthUnits="bytes" dfdl:lengthKind="explicit"
  dfdl:length="{ .. /header/InclLen }"/>
```

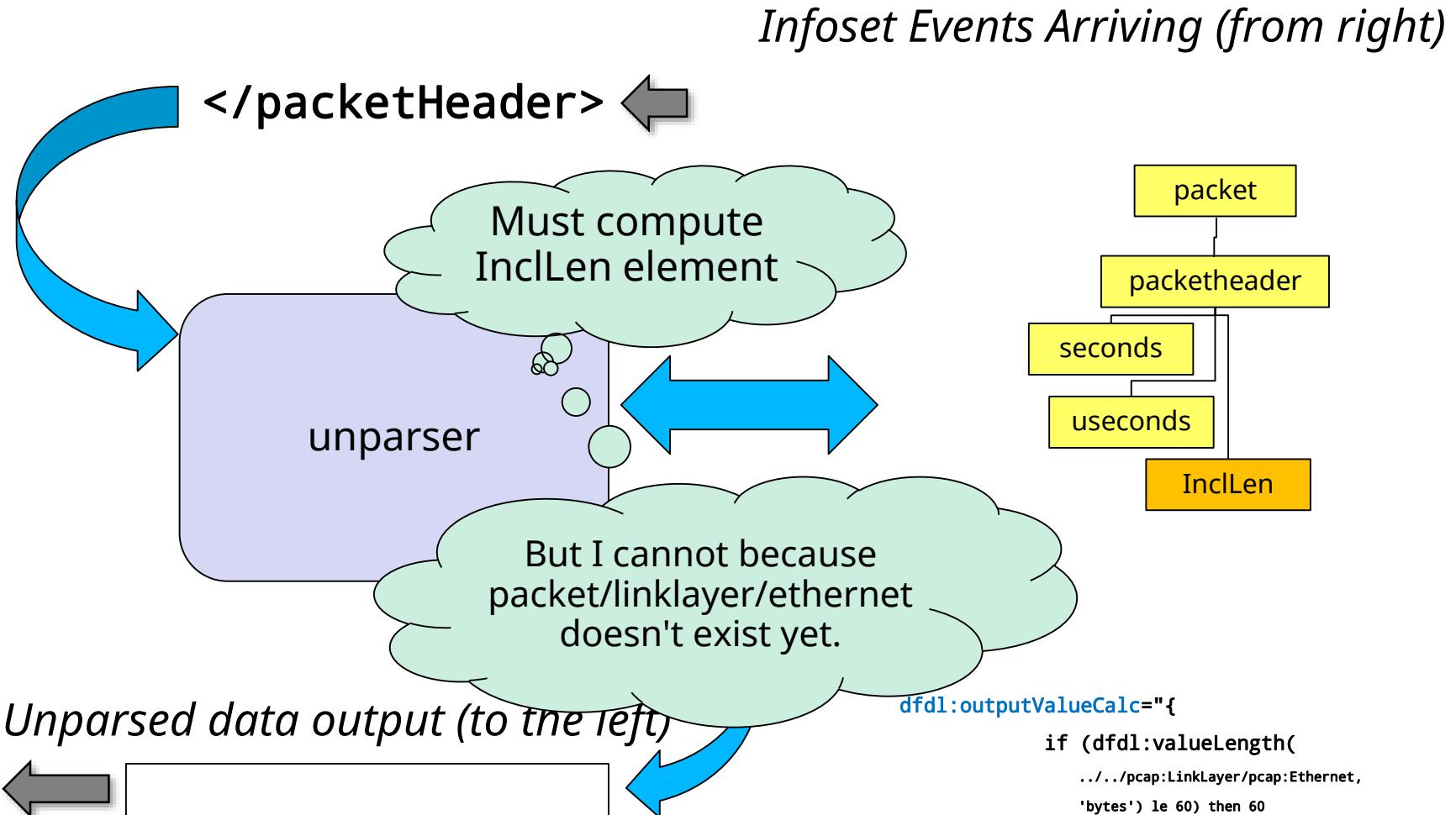
*Expressions
in DFDL's
expression
language
(like XPath)*



Streaming Unparsing Illustration



PCAP Streaming Unparsing Illustration

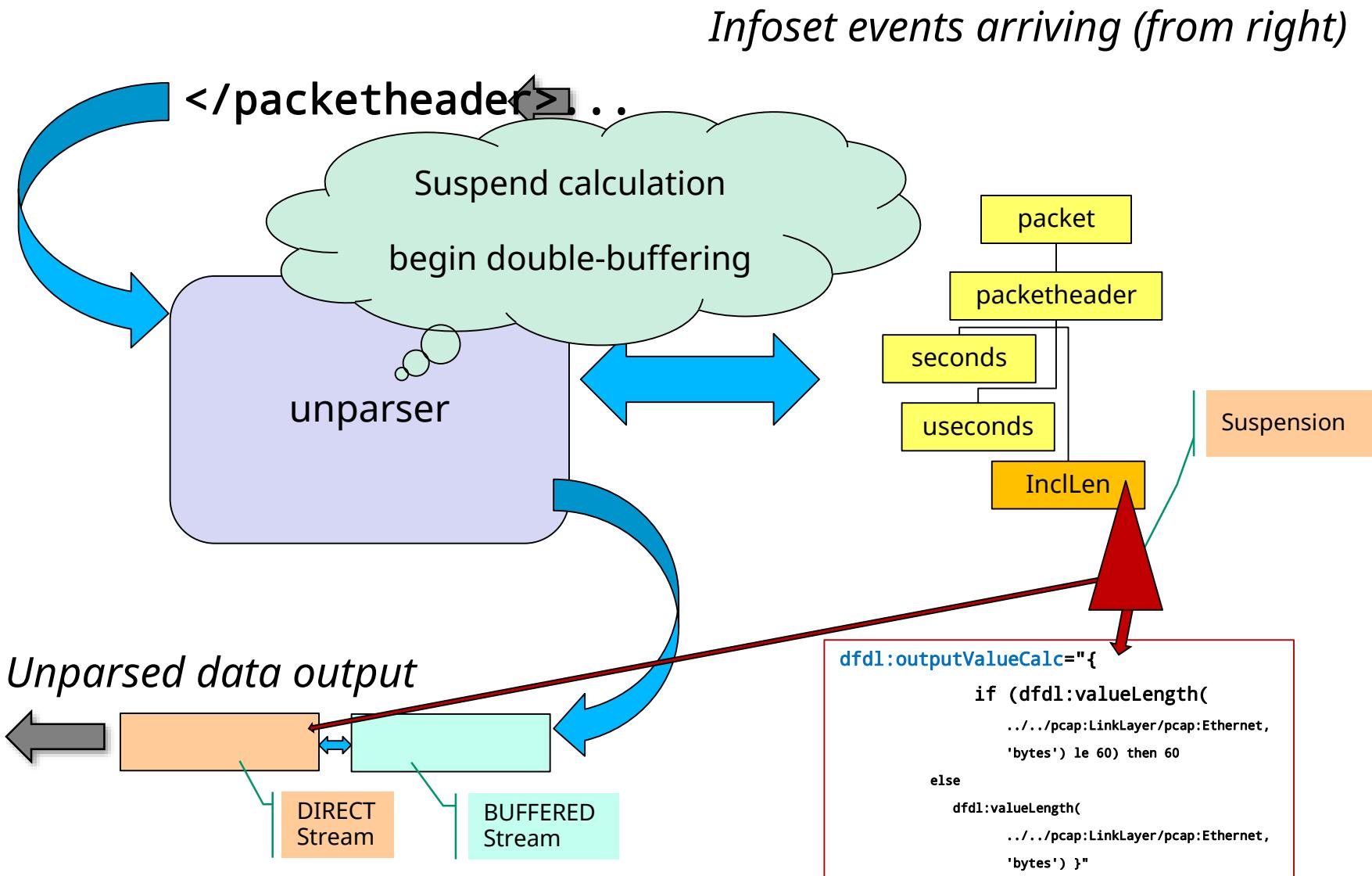


```
dfd1:outputValueCalc="{
    if (dfd1:valueLength(
        ../../pcap:LinkLayer/pcap:Ethernet,
        'bytes') le 60) then 60
    else
        dfd1:valueLength(
            ../../pcap:LinkLayer/pcap:Ethernet,
            'bytes') }"
```

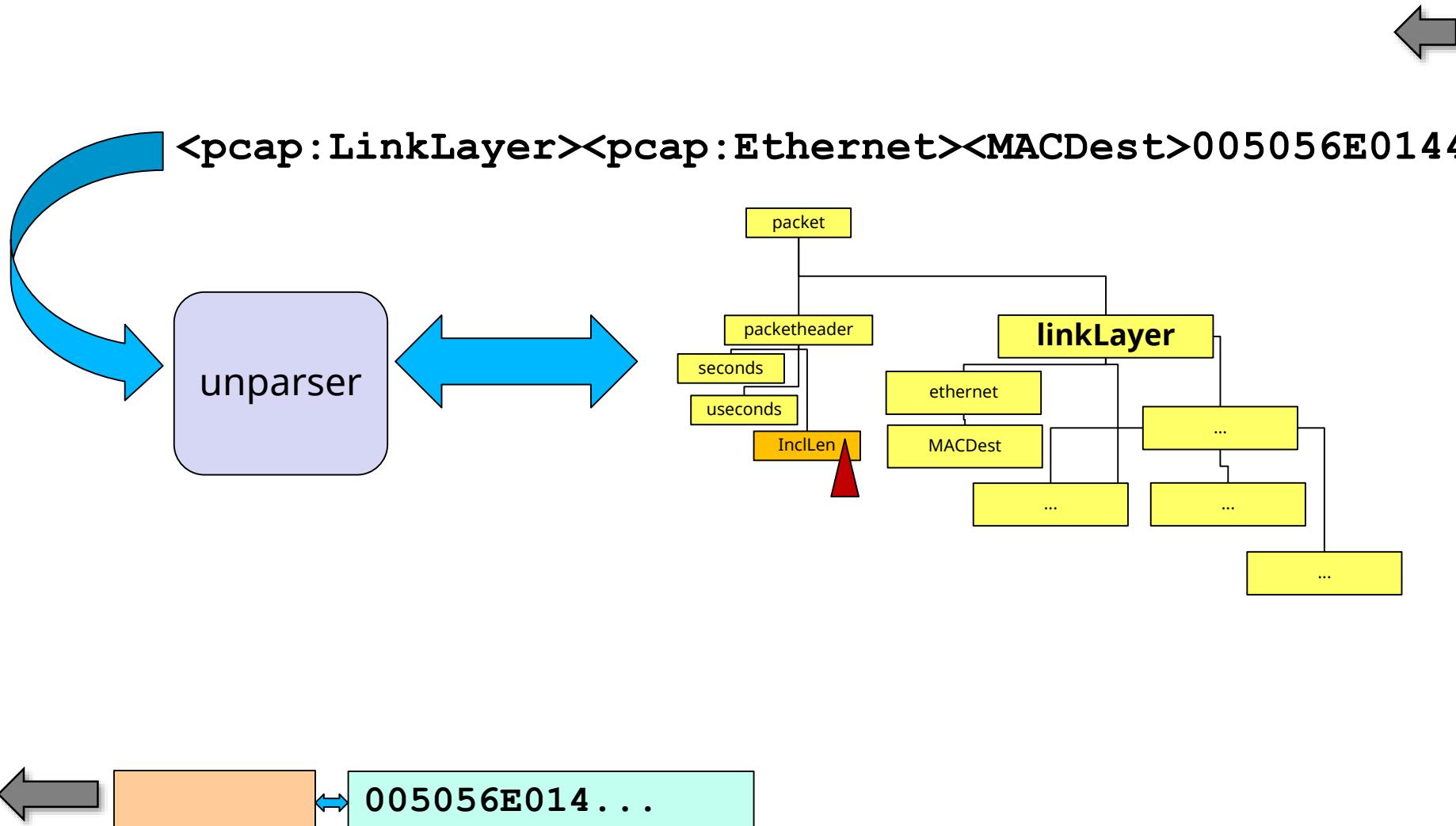
Suspensions and Double Buffering

- Must suspend `outputValueCalc` computation
 - Suspension object - like a co-routine object
- Blocked until packet/linklayer/ethernet
 - Event arrives
 - Added to InfoSet tree
 - All its sub-events/elements arrive and are added
 - And... all of it is unparsed to determine the length of its representation

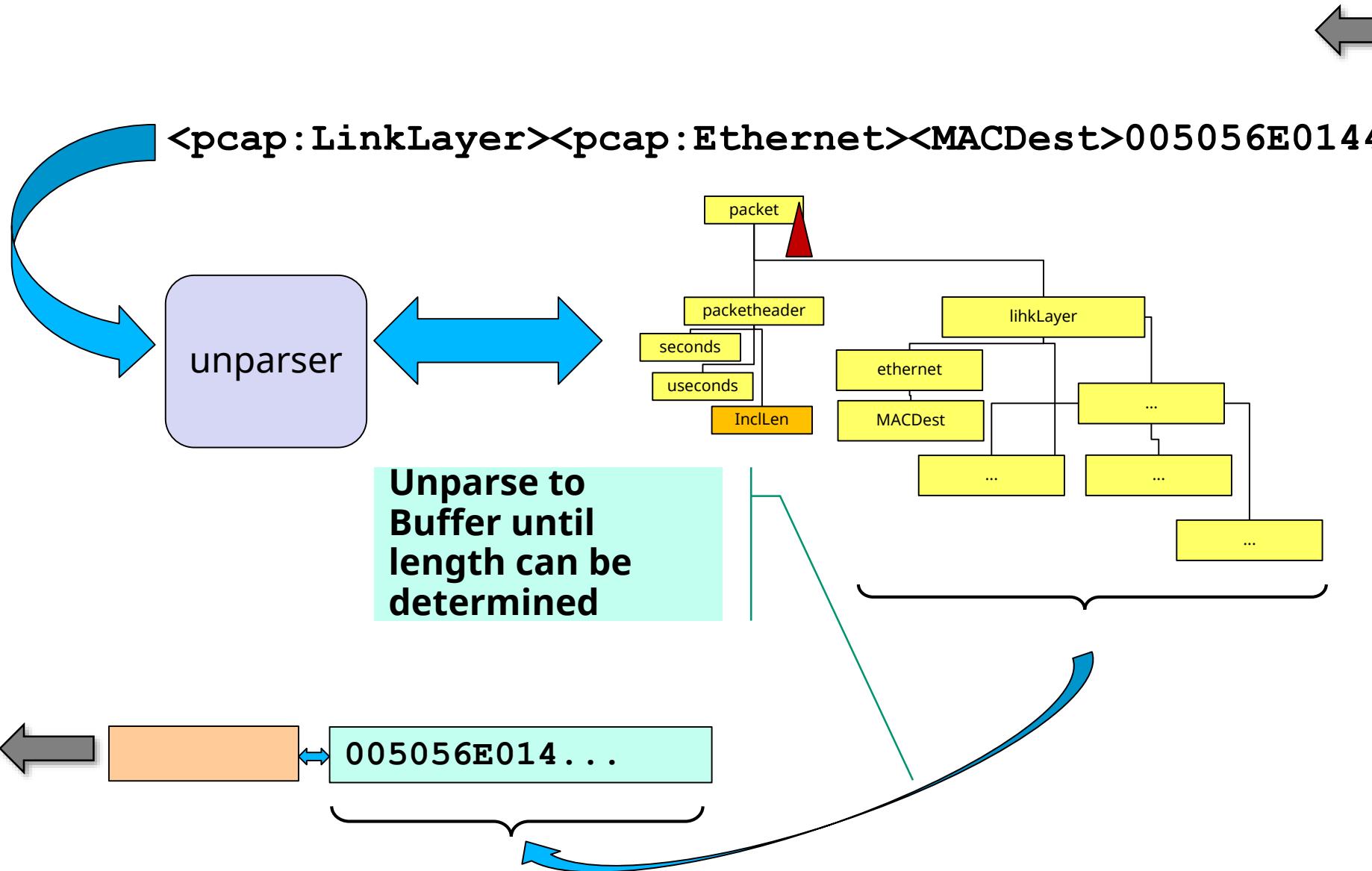
PCAP Streaming Unparsing Illustration



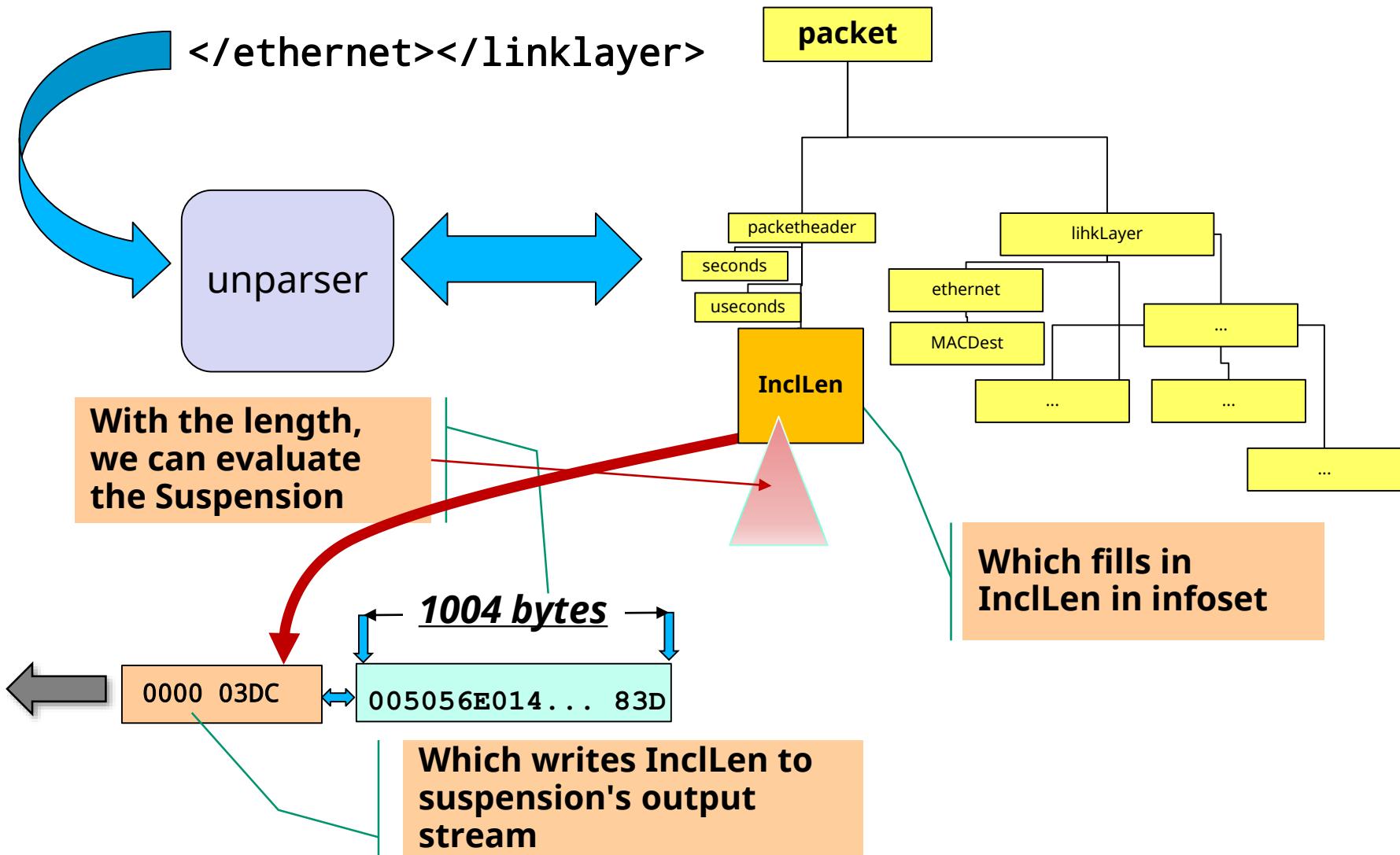
PCAP Streaming Unparsing Illustration



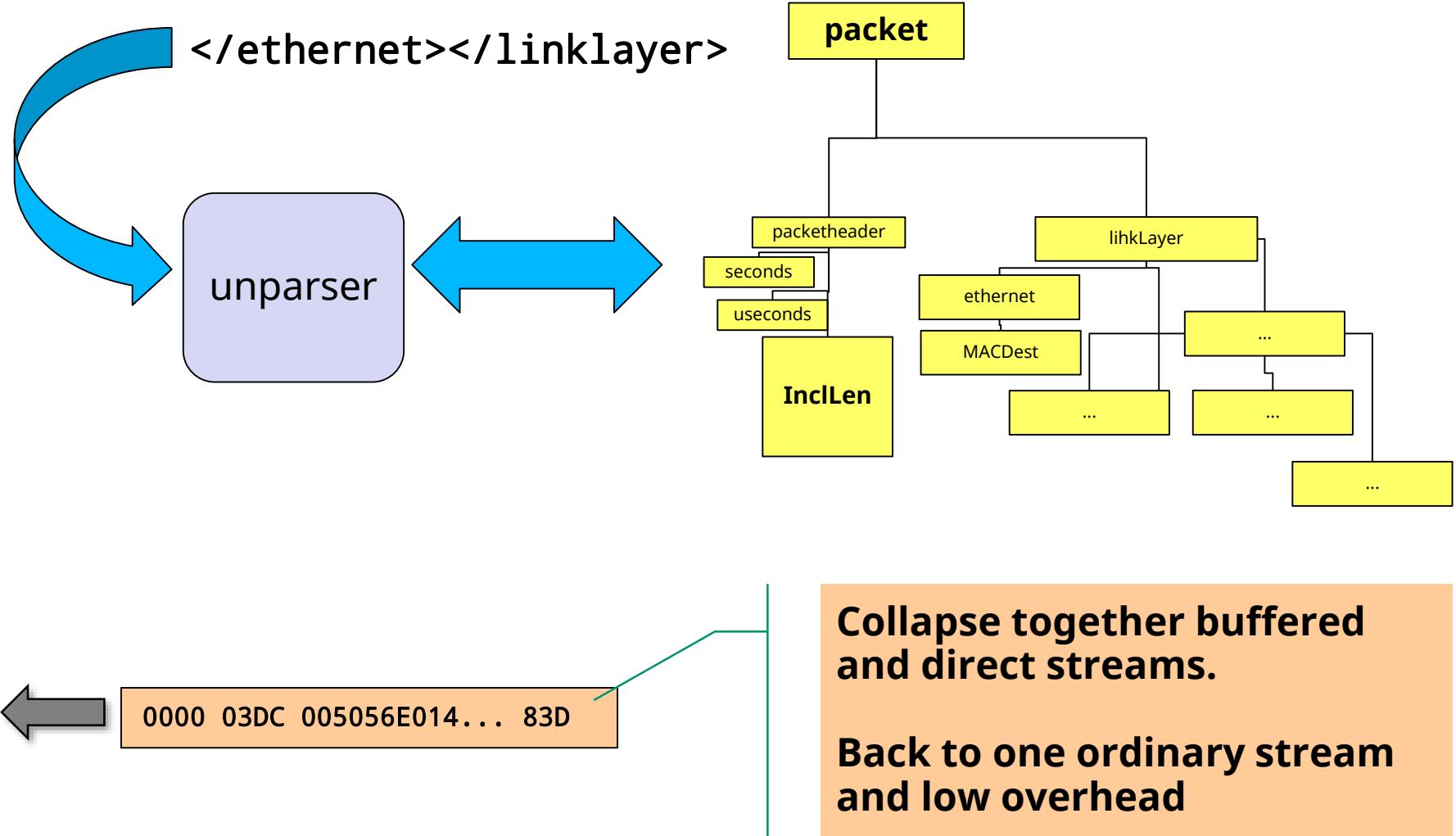
PCAP Streaming Unparsing Illustration



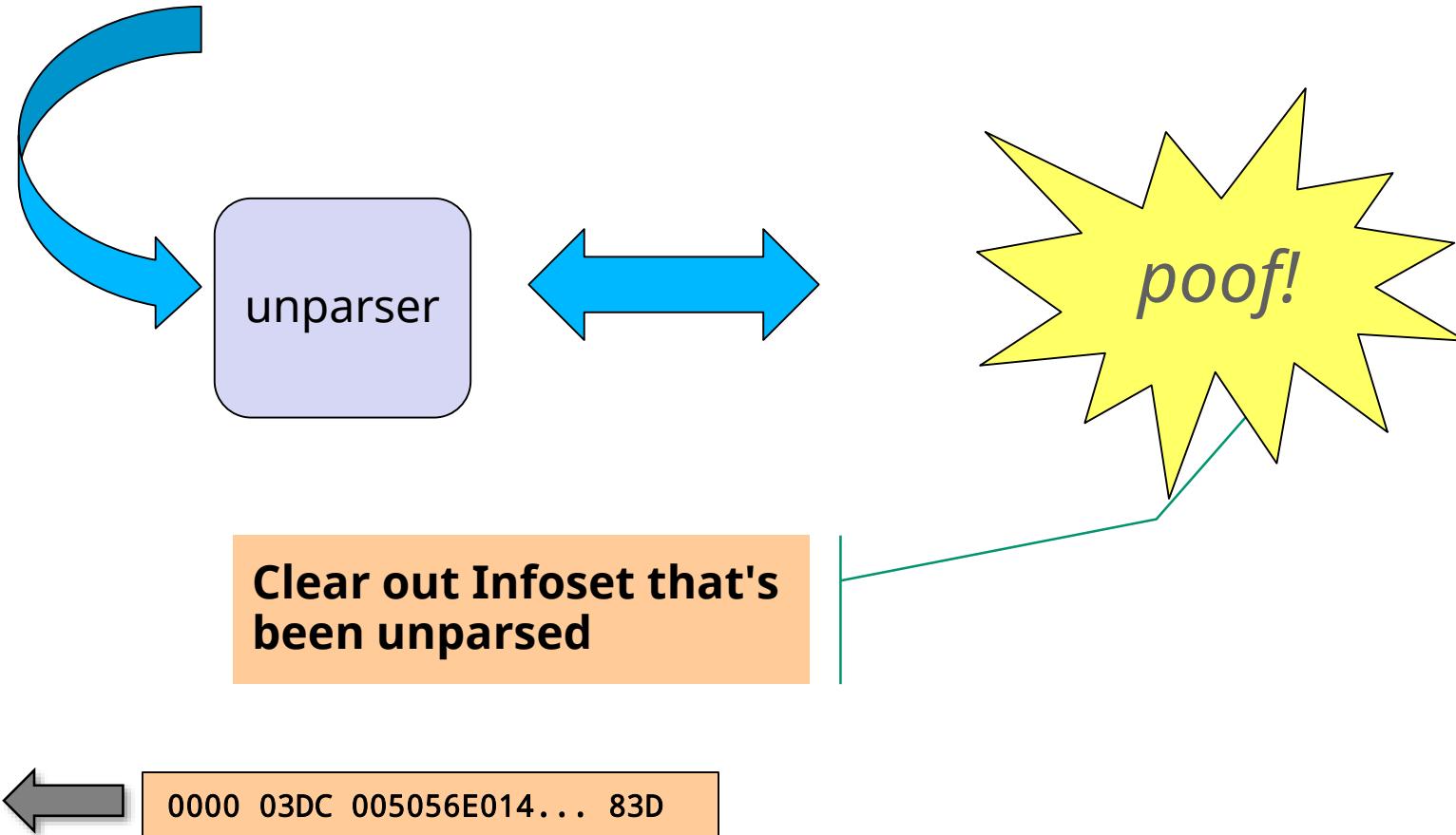
PCAP Streaming Unparsing Illustration



PCAP Streaming Unparsing Illustration



PCAP Streaming Unparsing Illustration



PCAP Streaming Unparsing Illustration

In summary

- Unparsing in DFDL is quite complex
- IMHO: Harder than parsing
- DFDL is much more expressive than other format description languages
 - dfdl:outputValueCalc - computed value elements
- Applications can truly be unaware of stored-lengths - leave that up to Daffodil!

DFDL Doesn't Do...

Limitations, Intended or Otherwise

Things DFDL (v1.0) Does

DFDL is for Data Sets

- Things typically thought of as files full of data about XYZ.
 - Rows, Tables
 - Header-Body-Trailer
 - Hierarchical / Nested record-oriented data
 - Messages
- Industry & Military data interchange formats
 - HL7, HIPAA-5010, SWIFT, NACHA, X25, Link16, etc.

Things DFDL (v1.0) Doesn't Do

- DFDL v1.0 was *not originally* intended for:
 - Document file formats
 - e.g., MS Office documents (.doc), or RTF, PDF
 - Archives like zip files or tar files
 - Core dump/memory image format
 - Storage format of a RDBMS table
 - Graphs of pointers - object graphs dumped from memory

Current DFDL v1.0 Language Limitations

Recursive types

- DFDL v1.0 *not* a *Turing-Complete* language
- On purpose - it's a feature, not a bug



Position of elements "by offset"

- Random jumping around data
- Ex: TIFF file format
 - TIFF cannot be described in DFDL v1.0

Thanks to
<http://langsec.org/occupy/>

Why is DFDL Needed?

There are *hundreds* of ad-hoc data format description systems

Every Enterprise Software Company

- IBM (10+ years)
- Oracle(10+ years)
- SAP(10+ years)
- Microsoft (10+ years)
- SAS
- Informati
- SyncSort
- AbInitio
- Pervasive
- Qlik/Expr
- Pentaho
- Dozens more

Every kind of software that takes in data:

Software world is very different now.

All this proliferation of redundant engineering can be avoided in the future.

DFDL Language Standard + 300K lines of Apache Licensed Daffodil can eliminate this data format problem **finally**.

- **proprietary**
- ***ad-hoc***
- ***incompatible***

Even within products of the same company!

Collaborators Needed

- Apache Daffodil (Incubating)
 - Growing our Community
 - To graduate from incubator to full Apache project status
requires committers from more organizations
 - Scala programmers wanted
 - Another mentor or two....
- DFDL Schemas
 - Join github DFDLSchemas community

Conclusion



Daffodil and DFDL

We can finally end the data format problem

DFDL

Basics

Innovations: calculated elements

What it can and cannot do (yet)

Daffodil



Big Scala code base contains - Compiler, Runtime, Test system, many tests

Runtime - Unparser with calculated elements requires sophisticated co-routine-style behavior

Questions?

DFDL Specification:

<http://ogf.org/dfdl>

DFDL Schemas:

<https://github.com/DFDLSchemas>

Daffodil Open Source:

<https://daffodil.apache.org>

Knarly (Gnarly?) Data BoF
Tonight 7pm (19:00)
Terrasse Room

Apache®, Apache Spark, Apache NiFi, ApacheCon, Apache Daffodil, and the Apache Feather Logo are either registered trademarks or trademarks of the [Apache Software Foundation](#) in the United States and/or other countries.

IBM®, InfoSphere are trademarks of the IBM Corporation.
ESA and DFDL4S are trademarks of the European Space Administration
SoftwareAG and webMethods are trademarks of SoftwareAG

END

Extra slides may follow for use in optional discussion.

Why is DFDL Needed?

Q But what about...

- Apache Avro
- Apache Thrift
- Google GPBs
- ASN.1 BER (or PER/DER/XER)

A Those are great, but are *prescriptive*.

They don't describe formats, they *are* data formats themselves.

We need a *descriptive* language.

Things DFDL (v1.0 + *BLOB*) Does

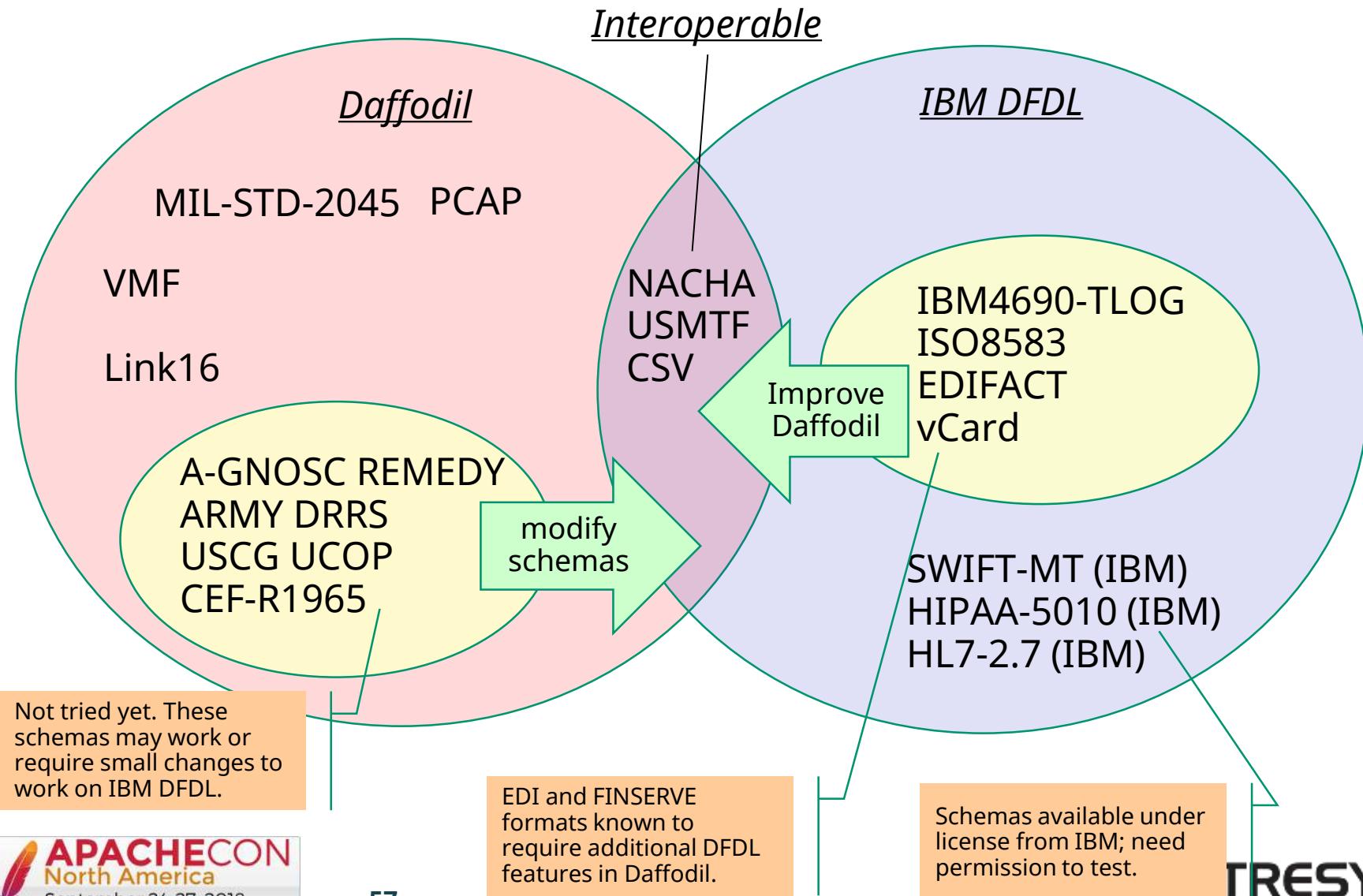
DFDL is for Images and Video

- Originally not in scope
- Large user demand to use DFDL on the metadata content of image file formats
 - Cybersecurity applications
- Adding BLOB (Binary Large Object) feature to DFDL language to enable DFDL to describe image files

Daffodil Future Development

- Cross-validation/test with IBM DFDL
 - Interop test on all IBM-created DFDL schemas
- Tutorials on writing/debugging DFDL schemas
- Improved trace and debug
- Full SAX-style streaming behavior for parsing, unparsing
- Faster schema compilation for large schemas
- Integrate into more frameworks
- Extensions: recursion, BLOB/CLOB, table-lookup,...

Interoperability – Daffodil & IBM



Why is DFDL Needed? - ASN.1 ECN

What about ASN.1 Encoding Control Notation?

- Already an ISO Standard (since 2008)
- Conceptually similar
 - Logical schema language + notations for physical representation
- Very different in the details.
- Developers [Love | Hate] [ASN.1 | XML]

Differences that matter:

- ASN.1 ECN
 - No open source implementation (as of 2018-08-29)
 - Extension of a binary data standard ASN.1 BER/PER/DER
 - Goal to describe legacy protocol messages
- DFDL
 - Open source Daffodil implementation
 - Extension of a textual data standard XML
 - Goal to be union of data integration tool capabilities for format description