Bullpen: NDL 1.0: A Primer

John Chapman

August 1, 2008

Is there a simple way to sort out the twisted interconnectivity of modular cable modem termination systems (M-CMTSs), fiber nodes and DOCSIS 3.0 service groups? What about adding asymmetrical downstream and upstream directions to the mix?

Trying to map out those components is what led me to create a Network Design Language (NDL), which I introduced in a paper at this year's Cable-Tec Expo. It's a shorthand, algebraic-based notation for describing a network entity, such as a router, a CMTS or another other network component or group of components.

NDL is designed both to decompose complex systems into more tractable sub-units and to assemble and map those defined sub-units onto actual sentences/formulas that represent the network or component. The intent is to enable anyone involved in any aspect of network design or deployment to easily represent and understand complex systems.

Those wanting a full explanation should refer to the paper. What follows is a working overview of NDL's three components: objects, ports and attributes.

Objects

NDL treats network components as objects. A base-level NDL object is intended to be some sort of field replaceable unit (FRU) such as a line card in a chassis. Examples include routers, switches and CMTSs, but may also include transmission elements such as splitters or fiber nodes.

For example, a generic chassis (CH) that has eight line cards, would be represented as:
CH = 8 * LC

An algebraic expression is used to indicate that eight line cards equal one chassis. Note how a short algebraic equation can concisely describe how a larger object is composed of smaller objects. The next sections will show that if the equation contained the properties of the line card, the properties of the chassis could be calculated from this equation.

This example also illustrates how objects are hierarchical. That is, objects can be added to make bigger objects. As the level of hierarchy continues, the objects become more abstract in nature, but still represent a physical collection of network entities. Examples include:
- A service group, which is a collection of frequencies on a set of fiber nodes
- All the equipment in one rack, which may be operating as a unit
- All the equipment in an entire network

Ports

These NDL objects can be described by their physical connectivity, which is based upon their physical ports. Physical ports generally are ports that occupy space somewhere on a front or back panel and have a cable connected to them. Examples include: Gigabit Ethernet (GigE) port and HFC downstream or upstream port.

Port Lists are contained in curly brackets { } immediately following the object name. There are two variables used in describing a port. There is the port count that indicates how many of those ports exist for that object, and the port type that indicates the function of the port.

Let's change our earlier example from a generic chassis into an Ethernet switch that has 24 GigE ports on each line card. The line card definition would be:
Ethernet_LC {24 GE}

The port type "GE" is a reserved port type that refers to Gigabit Ethernet.

Now, let's use NDL to calculate how many ports exist at the chassis level. The generic chassis expression, "CH," is replaced by a more relevant name.
Switch { }
= 8 * Ethernet_LC {24 GE}
= Switch {8 * 24 GE}
= Switch {192 GE}

The equation starts declaring that a switch is being defined, but the port types and counts are not known. An algebraic equation is used to indicate that there are eight line cards with 24 GigE ports per line card. The multiplier "8" is distributed into the parenthesis. The result is a switch with 192 GigE ports.

Also note the formatting used. The original declaration is placed on the first line. The subsequent equal signs are lined up for readability, while multiplications are done in line. The final result is on the last line.

This style of formatting is chosen for readability. Even though everything could fit into one line, it is much more readable if the lines are short and repeated or if like objects are lined up in columns rather than spread out in rows.

Attributes

Each object or port may contain one or more attributes, which are uniquely identified by their units. Some examples include: 10 Gbps, SFP (single form-factor pluggable), 6 MHz, 500 HHP and channelization.

Attribute lists are contained in square brackets [ ] immediately following a port or object definition. For example, to indicate the data capacity of an Ethernet line card:
LC {24 GE [1 Gbps]}
= LC {24 GE} [24 * 1 Gbps]
= LC {24 GE} [24 Gbps]

In the first line of this example, the 1 Gbps attribute is attached to the port type, and, therefore, is interpreted as a per port attribute. In the second line, a mathematical expression is shown in the attribute list. In the third line, the final result is shown. Since the attribute list on the second and third line is after the closing curly bracket ("}"), the attribute list is considered as being attached to the object.

Any number of attributes could be attached to an equation. As a result, care has to be taken to not add so many attributes that readability is sacrificed.

Also, obvious attributes should be excluded. In this case, the 1 Gbps attribute is obvious because the port type is GE, which is known to imply 1 Gbps. It is only included because it makes the math operation more clear. The second line is obvious and could be excluded. The attribute list on the third line is actually useful as it shows the total port capacity of the switch.

There are many ways to have done the formatting. For readability, it works best for the port list to stand out.

Special cases

These three components–objects, ports and attributes–form the basis of NDL, but special cases quickly arise.

For instance, NDL is able to handle unidirectional ports and asymmetrical bandwidth as is required by a CMTS object. NDL can also handle channelization of ports and line card redundancy. The attribute list is the way to handle variables such as different egress and ingress data rates.

An example would be:
CMTS { }
= (7 + 1 R) * DOCSIS_LC {5 x 10 [2 ch]}
= CMTS {35 DS, 70 US [2ch]}

The use of a mathematical formula to represent a system provides precision, simplicity and scalability. NDL is a self-documenting design language that can accommodate a wide range of special cases. Too many details, however, could impede conciseness.

The trick is always to retain readability so the end result is useful.

John Chapman is Cisco Fellow and chief architect, Cable Business Unit, for Cisco Systems. Reach him at jchapman@cisco.com. Drawn from a paper presented at the SCTE Cable-Tec Expo. More info at www.johntchapman.com.