A Simplified Formal Specification for BGP

Dan Pei[1], Dan Massey[2] and Lixia Zhang[1]

UCLA[1]
{peidan,lixia}@cs.ucla.edu

Colorado State University[2]
massey@cs.colostate.edu

UCLA CSD Technical Report TR040044

November 8, 2004

Abstract

As with those of other protocols, BGP protocol standard[1] specifies the protocol details using language description. However, language description can be ambiguous. In this technical report, we present a formal and short specification for BGP protocol such that most of the important BGP standard details are captured precisely using mathematic language. This report can be a quick and concise reference of BGP standard for those who have some basic BGP background.

In this report, we first define the several terminologies in Section 1. Then in Section 2, we define three categories of protocol functions at one router: operational environment, route filtering and aggregation policies, and path attribute manipulation policies. Then we use these functions to define the BGP protocol syntax and semantics requirements in Section 3.

1 Terminology Definitions

- AS Number: a 2-octet non-negative integer
  IP address: a 4-octet non-negative integer
  IP prefix: IP address/netmask, such as 131.179.0.0/16. Define $ipp_1 \leq ipp_2$ iff ipp_1 is included by ipp_2.

- BGP path attribute set: attributes=(origin, as_path, next_hop, med, local_pref, atomic_aggregate, aggregator, community), such that:

$origin$:	$\in \{IGP = 0, EGP = 1, INCOMPLETE = 2\}$
as_path :	is a sequence$(asseg_1, asseg_2, \ldots, asseg_n), n \geq 0$ such that: $\forall i, 1 \leq i \leq n$ $asseg_i = (AS\_SET = 1, \{a_1, \ldots, a_m\}), m \geq 1,$ $a_j =$ one AS Number$, 1 \leq j \leq m, where\ \{a_1, \ldots, a_m\}$ is an unordered set$, OR$ $asseg_i = (AS\_SEQUENCE = 2, (b_1, \ldots, b_l)), l \geq 1,$ $b_j =$ one AS Number$, 1 \leq j \leq l, where\ (b_1, \ldots, b_l)$ is an ordered sequence.
$next\_hop$:	IP address
med :	four-octet non-negative integer
$local\_pref$:	four-octet non-negative integer
$atomic\_aggregate$:	$\in \{0, 1\}$
$aggregator$:	AS Number:IP address
$community$:	a set of four-octet values.

2 Functions Given by Administrators and Vendors

BGP provides vendors and administrators with a great flexibility. The vendor may select from a broad range of implementation options and the administrator can further refine these options and add additional local policies. For example, the BGP specification imposes some constraints on the route selection process, but leaves most decisions up to the vendors and administrators. One BGP router may choose to select routes based on the shortest AS path length, but a second BGP router may ignore the AS path length altogether and select routes based on the some other local policies. Both routers fully comply with the BGP specification. In order to capture the full range of BGP in mathematical description, the following functions must be provided by the vendors and/or administrators.

These given functions fall into three categories. The first category lists the AS numbers for the router and its peers, the corresponding IP addresses, and other local network topology information. The second category lists the route filtering and aggregation policies for the router. The third category lists policies for updating individual BGP path attributes.

2.1 Operational Environment

For a BGP router b, we are given the following:

- $Peers(b) = \{0, ..., P\}$: a set of potential BGP peer routers. In our notation, Peer 0 will be used to represent static routes and routes learned from IGP.
- $as(x)$: the AS Number of x, $\forall x \in Peers(b) \cup \{b\}$. Define as(0)=as(b).
- $interfaces(x)$: a subset of x's interface IP addresses such that they are known by b. Define $interfaces(0) = interfaces(b)$.
- $ip(p)$: the IP address of p that is used in the peering session between b and p, where $ip(p) \in interfaces(p), \forall p \in Peers(b) - \{0\}$.
- $ipb(p)$: the IP address of b that is used in the peering session between b and p, where $ipb(p) \in interfaces(b), \forall p \in Peers(b) - \{0\}$.
- Function $same\_subnet(addr_1, \ldots, addr_n) \rightarrow \{0, 1\}$ such that $addr_1, \ldots, addr_n$ are IP addresses and $same\_subnet(addr_1, \ldots, addr_2) = 1$ iff $addr_1, \ldots, addr_n$ all share the same subnet.
- Function $multihop(b, p) = 0$ iff $\forall addr_1 \in interfaces(b), addr_2 \in interfaces(p)$, $same\_subnet(addr_1, addr_2) = 0$.
- EBGP(p): $\rightarrow \{0, 1\}$ such that EBGP(p)=1, iff $as(b) \neq as(p)$
- Internalcost(addr)= the IGP cost from b to $addr$, which is an integer, and maximum value is $max\_internal\_cost$.
- Length(aspath)=the length of aspath.
- MinRouteAdver(p): the time between two successive advertisements(from b to p) of routes to a particular destination
- MinRouteOrig(p): the time between two successive advertisements(from b to p) of routes to a destination in local AS

2.2 Route Filtering and Aggregation Policies

- Function $filter(p, direction, nlri, attributes) \rightarrow \{0, 1\}$ such that
  - $p \in Peers(b)$
  - $direction \in \{IN, OUT\}$
  - $nlri$ is an IP prefix
  - $attributes$ is a set of BGP attributes
  - $filter(p, IN, nlri, attributes) = 1$ iff the route $(nlri, attributes)$ will be accepted from peer p
  - $filter(p, OUT, nlri, attributes) = 1$ iff the route $(nlri, attributes)$ will be advertised to peer p
- $aggregates(b) =$ a set of IP prefixes to be aggregated by b
- Function $aggregate(agnlri, p, nlri_1, attributes_1, ..., nrli_n, attributes_n) \rightarrow (attributes, include, exclude)$ such that
  - $agnlri \in aggregates(b)$
  - $p \in Peers(b)$
  - $\forall i, nlri_i \leq agnlri$, AND $nlri \in exclude$ or $nlri \in include$
  - $exclude \cap include = 0$
  - $\forall nlri_i \in include$, $attributes.med = attributes_i.med$
  - $\forall nlri_i \in include$, $attributes.nexthop = attributes_i.nexthop$
  - if $attributes.origin = IGP$, then $\forall nlri_i \in include$, $attributes_i.origin = IGP$
  - if $attributes.origin = EGP$, then $\exists nlri_i \in include$ such that $attributes_i.origin = EGP$ and $\forall nlri_i \in include$, $attributes_i.origin \in \{EGP, IGP\}$
  - if $attributes.origin = INCOMPLETE$, then $\exists nlri_i \in include$ such that $attributes_i.origin = INCOMPLETE$
  - $attributes.community =$ any set of community values
  - $attributes.aggregator = 0$, or $as(b) : addr$ such that $addr \in interfaces(b)$
  - $attributes.atomic\_aggregate = 1$ if $\exists nlri_i \in includes$ such that $attributes_i.atomic\_aggregate = 1$
  - attributes.med=any med value
  - attributes.local_pref=any local_pref value
  - $attributes.as\_path = (asseg_1, \ldots, asseg_n), n \geq 0$ such that
    - if $\forall nlri_i, nlri_j \in include, attributes_i.as\_path = attributes_j.as\_path$, then $attributes.as\_path = attributes_i.as\_path, nlri_i \in include$
    - if $a \in as\_seq$ such that $(AS\_SEQUENCE, as\_seq) \in attributes.as\_path$, then $\forall nlri_i \in include, a \in attributes_i.as\_path$
    - if $a \in as\_set$ such that $(AS\_SET, as\_set) \in attributes.as\_path$, then $\exists nlri_i \in include, a \in attributes_i, as\_path$

* if $a = a_i, 1 \leq i \leq m$, such that $(AS\_SEQUENCE, (a_1, a_2, ..., a_m)) = asseg_j$, then: $\forall b = a_k, i < k \leq m$ OR $b \in asseg_k, j < k \leq n$,
  if $b = b_x, 1 \leq x \leq s$, such that $(AS\_SEQUENCE, (b_1, b_2, ..., b_s)) = asseg_t$ AND $1 \leq t \leq u, (asseg_1, asseg_2, \ldots asseg_u) = attributes_w.as\_path$, $nlri_w \in include$, then $a = b_y, 1 \leq y < x$ OR $a \in asseg_z, 1 \leq z < t$.
* if $a = a_i, 1 \leq i \leq m$, such that (AS_SET OR $AS\_SEQUENCE$, $(a_1, a_2, ..., a_m)) = asseg_j$, then $a \neq a_k, \forall 1 \leq k \leq m, k \neq i$ AND $a \notin asseg_k, \forall 1 \leq k \leq n, k \neq j$.

2.3 Path Attribute Manipulation Policies

- In all of the following function definitions:
  - $p \in Peers(b), fp \in Peers(b)$
  - $nlri$ is an IP prefix
  - $direction \in \{IN, OUT\}$
  - $attributes$ is a set of BGP attributes
  - $function(0, OUT, nlri, attributes)$ is not defined.
- Function $setorigin(0, IN, nlri, attributes) \rightarrow \{0, 1, 2\}$ and
  setorigin(p>0, IN/OUT, nlri, attributes)=attributes.origin.
- Function $setmed(p, direction, nlri, attributes) \rightarrow$ non-negative integer
- Function $setlocalpref(p, direction, nlri, attributes) \rightarrow$ four-octet non-negative integer such that
  setlocalpref(p, OUT, nlri, attributes)=$\begin{cases} attributes.local\_pref, if\ as(b) = as(p) \\ \emptyset, otherwise. \end{cases}$
- Function $setcommunity(p, direction, nlri, attributes) \rightarrow$ a set of four-octet values such that
  setcommunity(p>0, IN, nlri, attributes)=attributes.community
  If $as(p) \neq as(b), attributes.med \neq 0$, then $setcommunity(p, OUT, nlri, attributes)$'s default value must be 0.
- Function $setaspath(p, direction, nlri, attributes) \rightarrow as\_path$ such that
  - $setaspath(0, IN, nlri, attributes) = \emptyset$, $setaspath(p > 0, IN, nlri, attributes) = attributes.aspath$
  - if $as(p) = as(b)$ then $setaspath(p, OUT, nlri, attributes) = attributes.aspath$
  - if $as(p) \neq as(b)$ then $setaspath(p, OUT, nlri, attributes) = (as(b))^+ \cdot attributes.aspath$
- Function $setnexthop(p, fp, direction, nlri, attributes) \rightarrow$ IP address, where fp is the the peer from which the route is learned, such that
  - $setnexthop(0, 0, IN, nlri, attributes) \in interfaces(b)$, $setnexthop(p > 0, fp = p, IN, nlri, attributes) = attributes.nexthop$.
  - if $as(p) = as(b)$ then $setnexthop(p, OUT, nlri, attributes) = attributes.nexthop$ OR $setnexthop(p, OUT, nlri, attributes) \in interfaces(b)$.
  - if $as(p) \neq as(b) AND multihop(p, b) = 0$ then $same_subnet(setnexthop(p, fp, OUT, nlri, attributes), ip(p)) = 1$ and one of:

* setnexthop(p, fp, OUT, nlri, attributes)=ipb(p) OR
  - subset(ip(p), addr)=1 AND $addr \in interfaces(b)$ AND $setnexthop(p, fp, OUT, nlri, attributes) = addr$
  - $as(fp) = as(b)$ AND $setnexthop(p, fp, OUT, nlri, attributes) \in interfaces(fp)$
  - $as(fp) \neq as(b)$ AND $setnexthop(p, fp, OUT, nlri, attributes) = attributes.nexthop$ and $\exists addr \in interfaces(b)$ such that $same_subnet(addr, ip(p)) = 1$

– if $as \neq as(b), multihop(p, b) = 1$ then either:

* $setnexthop(p, fp, OUT, nlri, attributes) = attributes.nexthop$
* $setnexthop(p, fp, OUT, nlri, attributes) = ipb(p)$.

3 Protocol Definition

This section will describe the protocol semantic definition and we assume that several protocol syntactic functions will take care of the tasks such as maintaining peering section, reporting peer status, checking the format of incoming update packets, formating the outbound update packets.

3.1 Protocol Syntactic Functions and Environment

- T=the current time, and it is a integer.
- Peering Session Function:

$$UP(p > 0, T) = \begin{cases} 1, \text{peering session between } b \text{ and } p \text{ is in ESTABLISHED STATE at time T,} \\ 0, otherwise. \end{cases}$$

  Define UP(0, T)=1. Up(p,T) also maintains the peering session, including sending /processing OPEN/KEEPALIVE/NOTIFICATION messages.

- BGP route update=$(wnlri, anlri, attributes, time)$, such that
  $wnlri$: a set of IP prefixes of the routes to be withdrawn
  $anlri$: a set of IP prefixes of the routes to be announced($wnlri \cap anlri = \emptyset$)
  $attributes$: BGP path attribute set.
  $time$: the time the route is received or sent
- Any changes of the routes learned from Peer 0 (static routes or routes learned from IGP) will be translated to a BGP update format, applying any applicable given functions.
- Last_Recv(p,T)=the last UPDATE message that b received from p prior to time T
- Time T will increase by 1 after any of the following:
  - Given Functions changed(including policy changes)
  - receiving an update from a peer
  - UP(p,T) function changes

3.2 Protocol Semantic Definition

These functions are conceptual, the implementation may choose any algorithm as long as they conform to these definitions.

- Import(p,nlri,attributes) → path attributes such that attributes=

$$\begin{cases} \emptyset, & if\ as(b) \in attributes.as\_path; \\ \emptyset, & if\ filter(p, IN, nlri, attributes) = 1; \\ \{setorigin(p, IN, nlri, attributes), \\ setaspath(p, IN, nlri, attributes), \\ setnexthop(p, p, IN, nlri, attributes) \\ setmed(p, IN, nlri, attributes), \\ setlocalpref(p, IN, nlri, attributes), \\ attributes.atomic\_aggregate, \\ attributes.aggregator, \\ attributes.community\}, & otherwise \end{cases}$$

- Select(p,attributes): → real number such that:
  - Select(p,attributes)=0, iff RIB-Local(attributes.nexthop,T)= $\emptyset$
  - Select(p,attributes) =attributes.local_pref $+(1 - \frac{Length(attributes.aspath)}{max\_aspath\_length+1}) + (1 - \frac{attributes.origin}{max\_origin+1}) * 10^{-1} + (1 - \frac{attributes.med}{(max\_med+1)}) * 10^{-2} + \frac{EBGP(p)}{2} * 10^{-3} + (1 - \frac{Internalcost(attributes.nexthop)}{(max\_internal\_cost+1)}) * 10^{-4} + (1 - \frac{router\_id}{(max\_router\_id+1)}) * 10^{-5}$

- Export Policy function:$\{p, fp, nlri, attributes\}$ → path attributes such that Export(p,fp,nlri, attributes)=

$$\begin{cases} \emptyset, & if\ ip(p) = attributes.next\_hop; \\ \emptyset, & if\ filter(p, OUT, nlri, attributes) = 1; \\ \{setorigin(p, OUT, nlri, attributes), \\ setaspath(p, OUT, nlri, attributes), \\ setnexthop(p, fp, OUT, nlri, attributes), \\ setmed(p, OUT, nlri, attributes), \\ setlocalpref(p, OUT, nlri, attributes), \\ attributes.atomic\_aggregate, \\ attributes.aggregator, \\ setcommunity(p, OUT, nlri, attributes)\}, & otherwise. \end{cases}$$

- Adj-RIB-In(p, nlri,T) → path attributes such that attributes=

$$\begin{cases} \emptyset, & if\ UP(p,T) = 0; \\ \emptyset, & if\ UP(p,T) = 1\ AND\ nlri \in Last\_Recv(p,T).wnlri; \\ Last\_Recv(p,T).attributes, & if\ UP(p,T) = 1\ AND\ nlri \in Last\_Recv(p,T).anlri, \\ AdjRIBIn(nlri, p, T-1), & otherwise. \end{cases}$$

- RIB-Local(nlri,T)=(fp, attributes)=(p_{best}, Adj-RIB-In($p_{best}, nlri, T$)), $p_{best} \in peers(b)$ such that: select(p_{best},Adj-RIB-In(p_{best},nlri,T))= $\max_{p \in peers(b)}$ {select(p, Import(p, nlri, Adj-RIB-In(p,nlri,T))} > 0.
- Adj-RIB-Out(p,nlri,T):→ path attributes such that
  - Adj-RIB-Out(p,nlri,T)=Export(p,RIB-Local(nlri,T).fp, RIB-Local(nlri,T).attributes) iff $nlri \notin aggregates(b)$ OR
  - Adj-RIB-Out(p,nlri,T)=agattributes, if $nlri \in aggregates(b)$ such that:
    - $aggregate(nlri, p, nlri_1, attributes_1, \ldots, nlri_n, attributes_n) = (agattributes, include, exclude)$, and
    - $|include| \geq 2$.
- setatomicaggregate(p,nlri,attributes)=
$$\begin{cases} 1, & if\ attributes.atomic\_aggregate = 1 \\ 1, & nlri \in aggregates(b), and\ \exists nlri_{in} \leq nlri, such\ that: \\ & \quad Adj-RIB-In(p, nlri_{in}, T) \neq \emptyset, AND Adj-RIB-Out(p, nlri_{in}, T) = \emptyset \\ 0, & otherwise \end{cases}$$
- Adj-RIB-Out change sequence $RibOutSeq(p, nlri, T)$ is a sequence of path attributes: $(attributes_1, \ldots, attributes_m)$ such that:
  - $\forall 1 \leq i \leq m, attributes_i = Adj-RIB-Out(p, nlri, t_i), t_1 < ... < t_m \leq T$, AND
  - $\forall 1 \leq i < m$, $attributes_i \neq attributes_{i+1}$ AND
  - $\forall 1 \leq i < m$, there is no Adj-RIB-Out(p,nlri) changes between t_i and t_{i+1} or between t_m and T.
- Advertisement sequence AdvSeq(p,nlri,T)=$(a_1, ..., a_n)$, such that:
  - a_i is the update that b sent to p, AND
  - $\forall i$, EITHER $nlri \in a_i.anlri$, OR $nlri \in a_i.wnlri$, AND
  - $a_1.time < \ldots < a_n.time \leq T$, AND
  - $\forall 1 \leq i < n$, there is no update a from b to p such that $nlri \in a.nlri$ AND: $a_i.time < a.time < a_{i+1}.time$ OR $a_n.time < a.time < T$

3.3 Requirements

$\forall p > 0, nlri, T$, $AdvSeq(p, nlri, T) = (a_1, ..., a_n)$ must satisfy:

R 1 *Advertisement Frequency Requirements:*

- *if* $as(b) \neq as(p)$, *then:* $\forall 1 \leq i < n$, *if* $nlri \in a_i.anlri$ *AND* $nlri \in a_{i+1}.anlri$, *then* $a_{i+1}.time - a_i.time \geq MinRouteAdver(p) * 0.75$.

- $\forall 1 \leq i < n$, *if* $a_{i+1}.attributes.aspath = \emptyset$, *then:* $a_{i+1}.time - a_i.time \geq MinRouteOrig(p) * 0.75$.

R 2 $\exists B$, *such that:*

- *if* $\forall i \leq B, Adj\text{-}RIB\text{-}Out(p, nlri, T) = Adj\text{-}RIB\text{-}Out(p, nlri, C-i) \neq \emptyset$, *then* $nlri \in a_n.anlri, AND\ a_n.attributes$ $Adj - RIB - Out(p, nlri, T).attributes$.
- *if* $\forall i \leq B, Adj\text{-}RIB\text{-}Out(p, nlri, T) = Adj\text{-}RIB\text{-}Out(nlri, C - i) = \emptyset$, *then* $nlri \in a_n.wnlri, AND$ $nlri \in a_{n-1}.nlri$.

R 3 *Sequence* $(a_1.attributes, \ldots, a_n.attributes)$ *must be a subsequence of RibOutSeq(p,nlri,T).*

References

[1] Y. Rekhter, T. Li, and S. Hares. Border Gateway Protocol 4. http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp4-26.txt, Oct 2004.