# A Lightweight Library for Building Scalable Tools

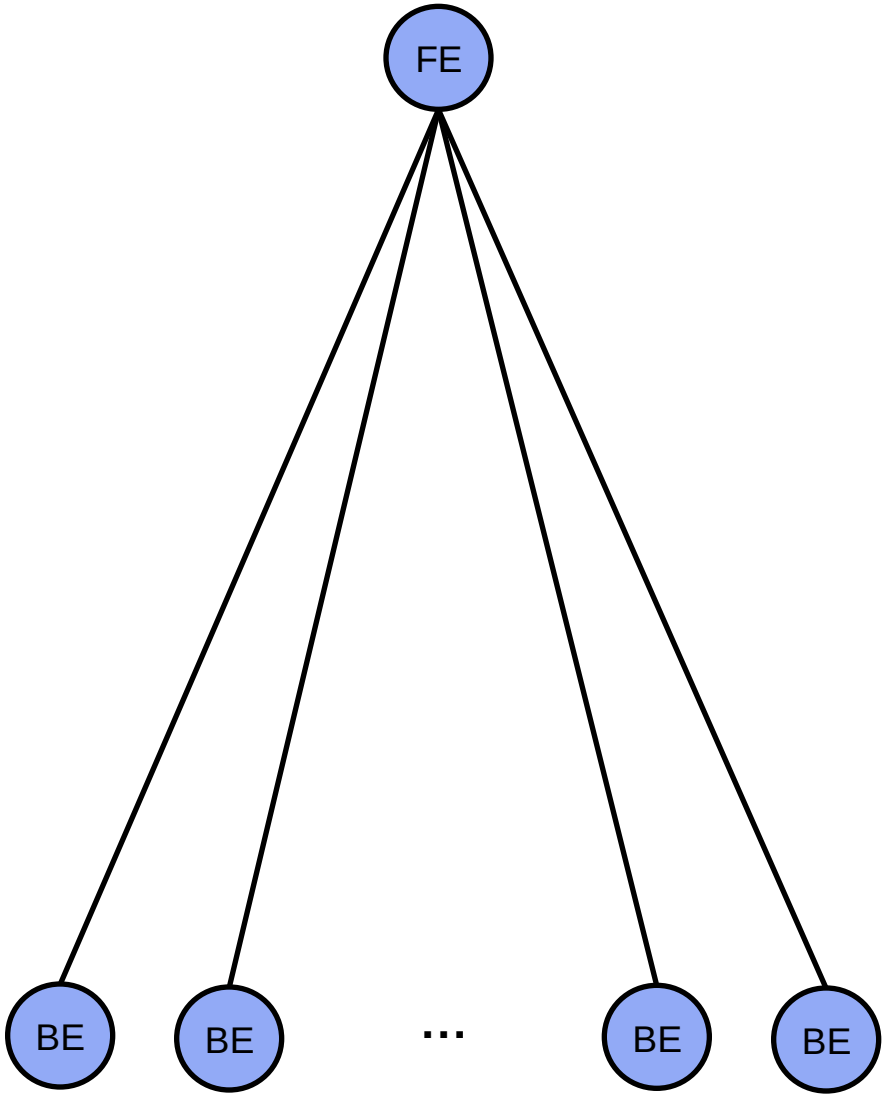
**Emily R. Jacobson**, Michael J. Brim, Barton P. Miller
Paradyn Project
University of Wisconsin
jacobson@cs.wisc.edu

June 6, 2010
Para 2010: State of the Art in Scientific and Parallel Computing

# MRNet Motivation

Example Tool:

Performance Analysis

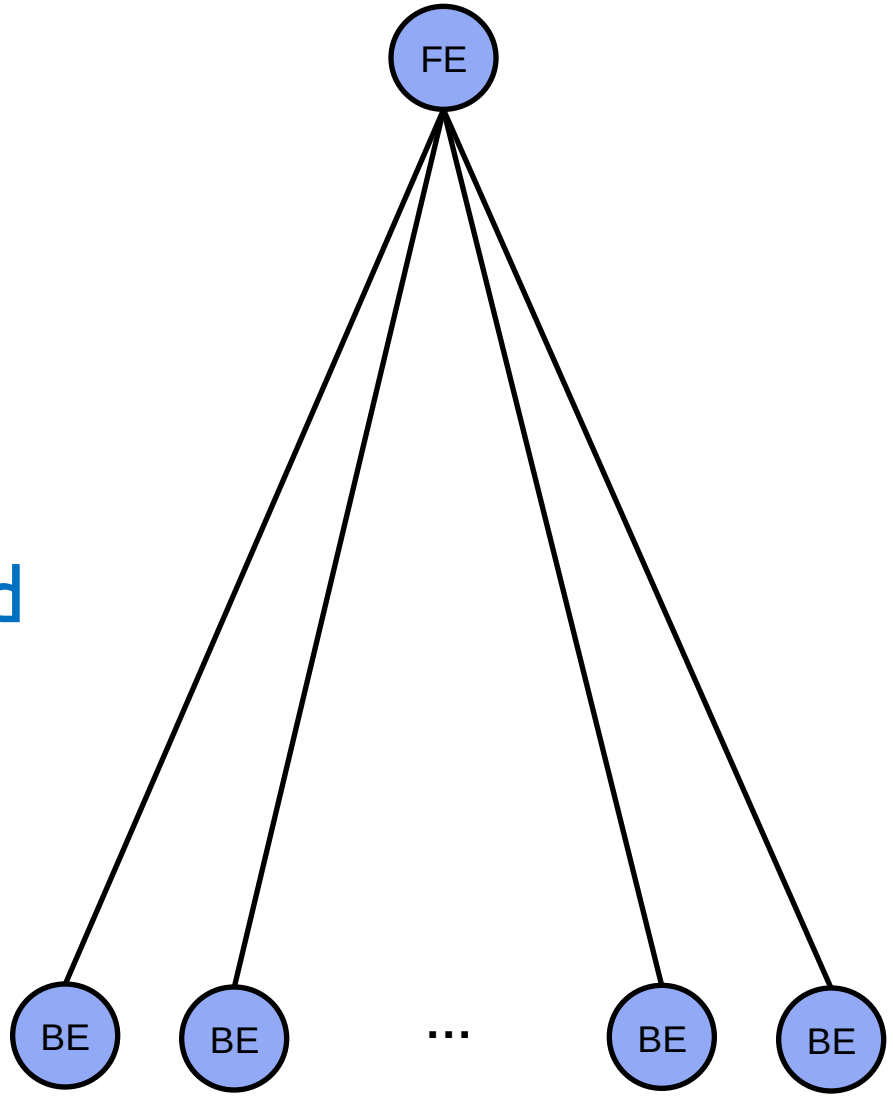


# MRNet Motivation

Example Tool:

Performance Analysis

As scale increases, front-end becomes bottleneck




100,000s of nodes

# MRNet Goals

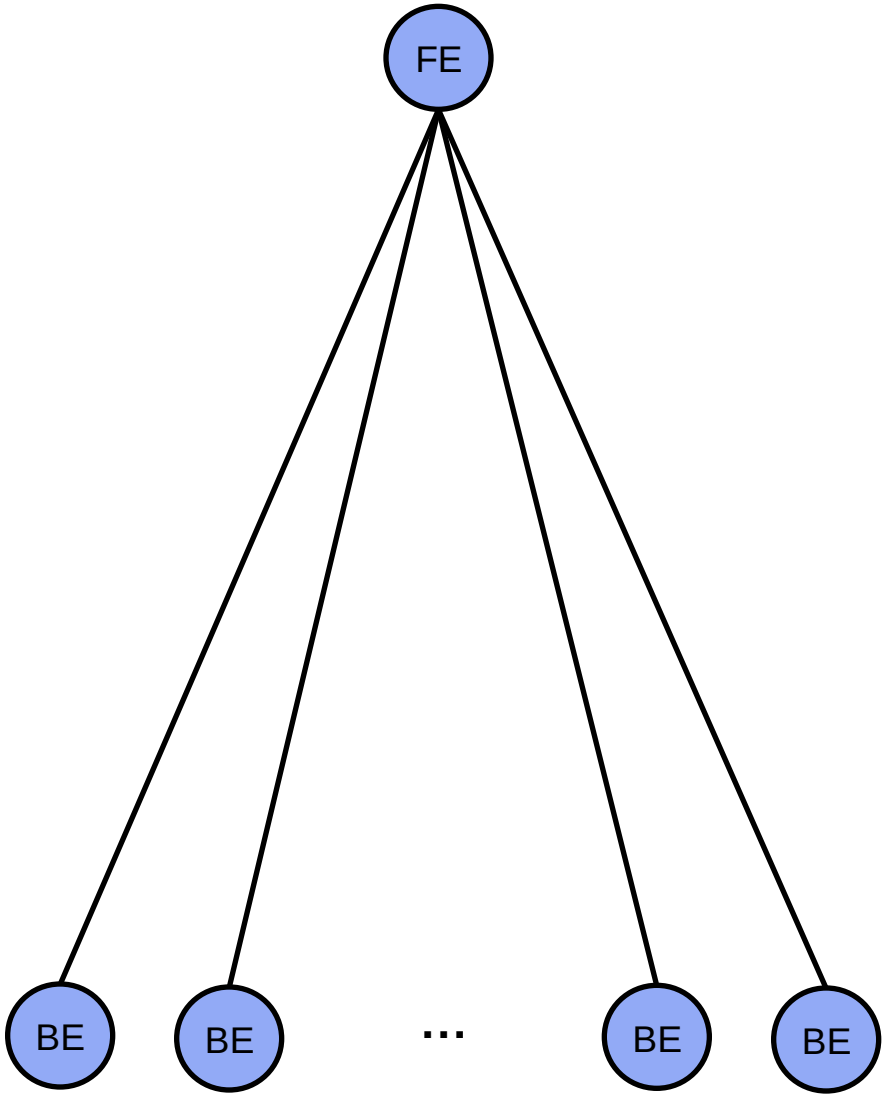
- Provide infrastructure for building tools that scale to the largest computing platforms

- Support scalability for command, computation, and data collection

# MRNet Features

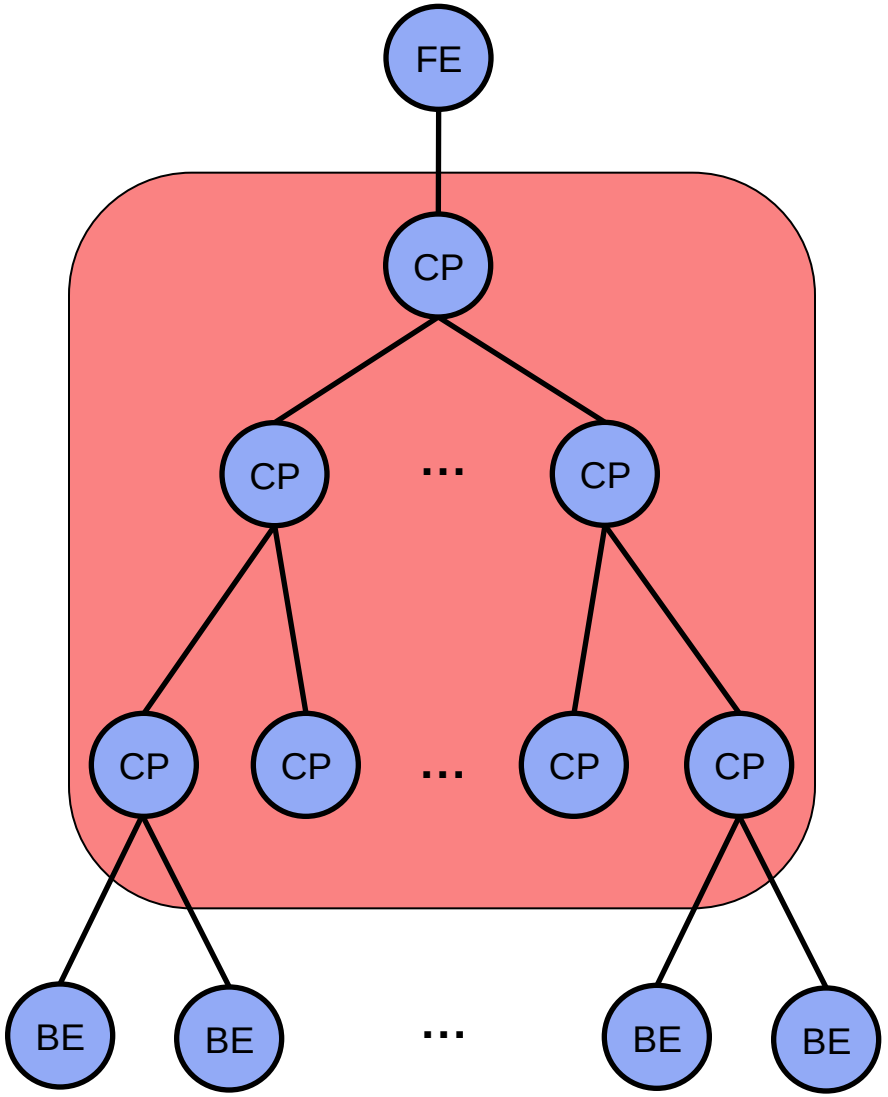
- **Scalable** – Handle 100,000's of nodes
- **Multi-platform** – Cray XT, IBM BlueGene, Linux clusters, AIX, Solaris, Windows
- **Reliable** – Automatic fault recovery
- **Flexible** – Target a wide variety of tools, applications, and architectures
- **Customizable** – Easily extend to new algorithms and requirements
- **Open Source**

# TBŌN Model



# TBŌN Model

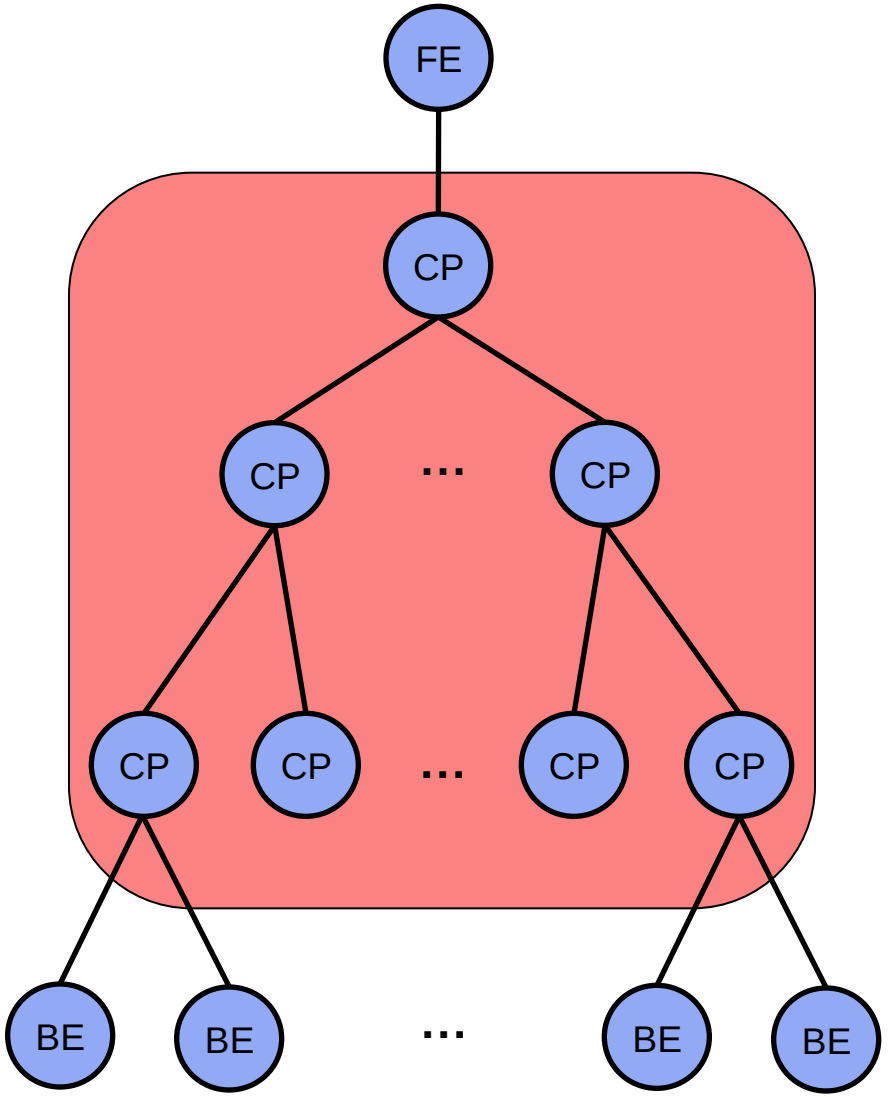
MRNet makes use of a software tree-based overlay network, TBŌN



# TBŌN Model

TBŌNs provide:

- Scalable multicast
- Scalable gather
- Scalable data aggregation

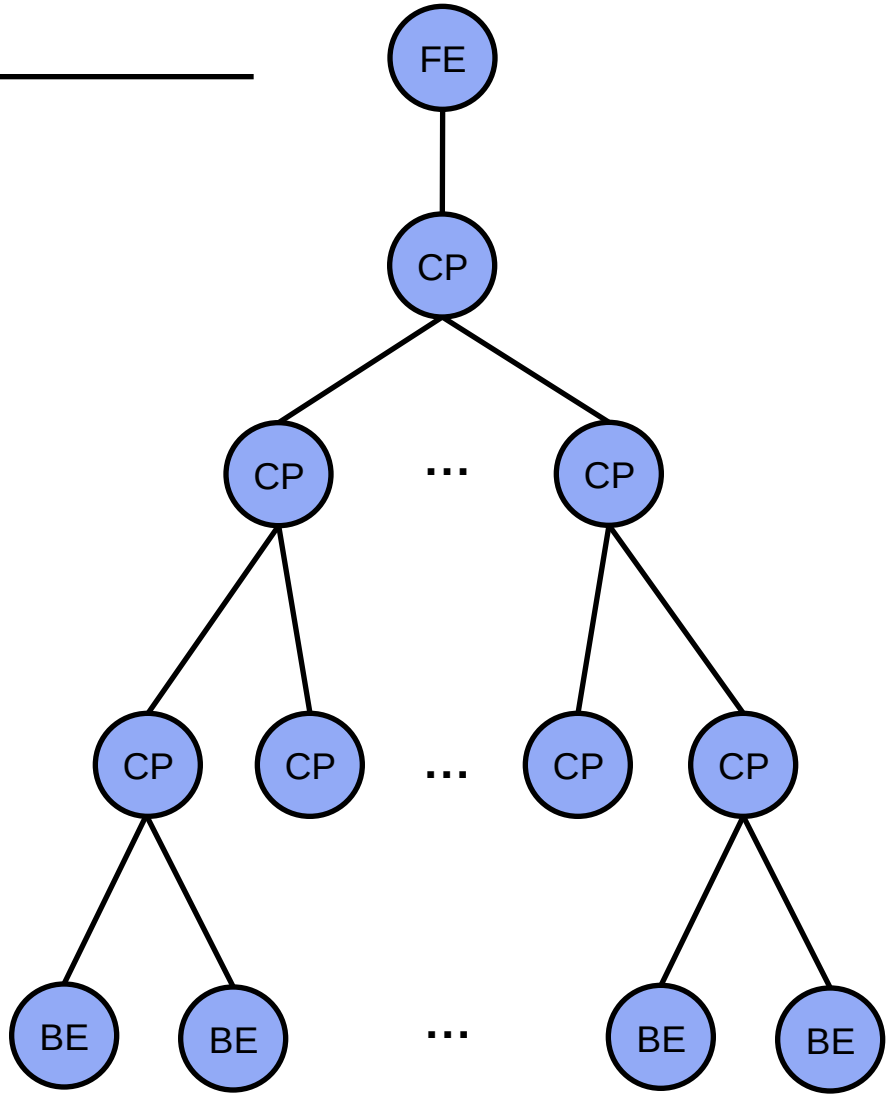


# TBŌN Model

Application Front-end

Tree of
Communication Processes

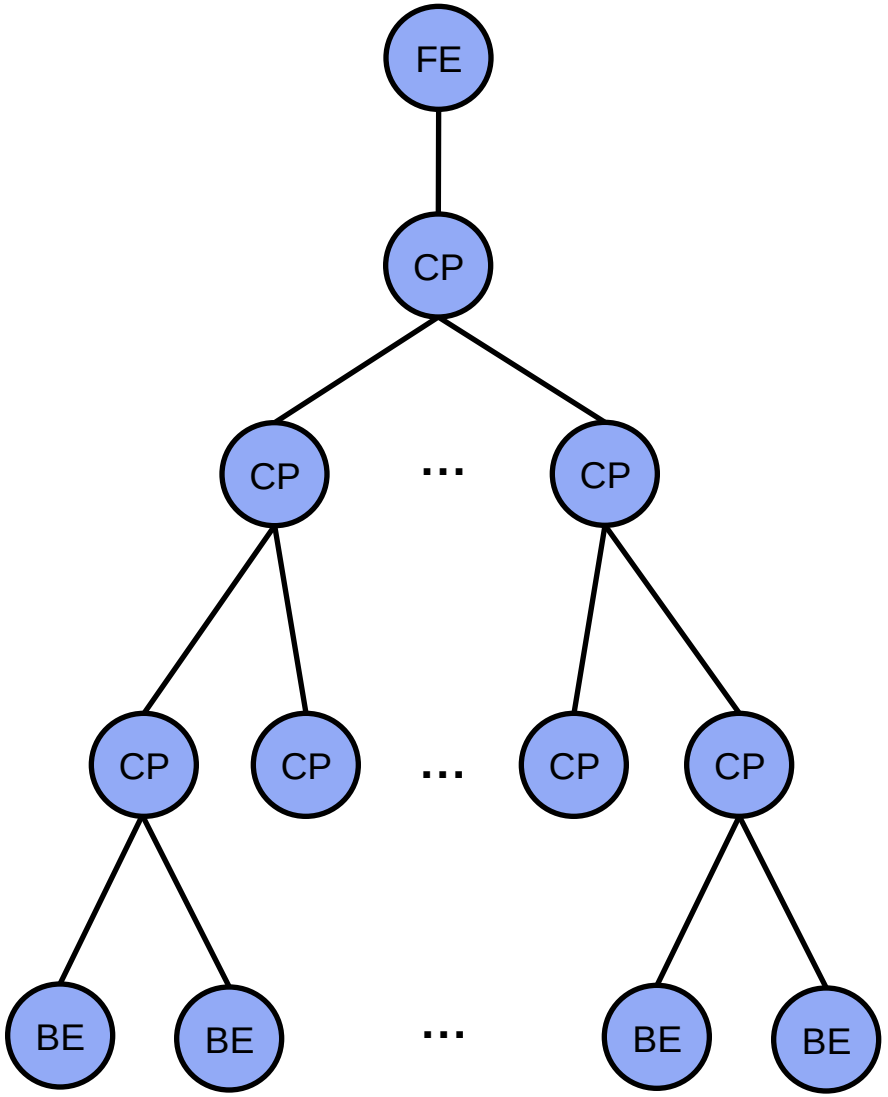
Application Back-ends



# MRNet: An Easy-to-use TBŌN

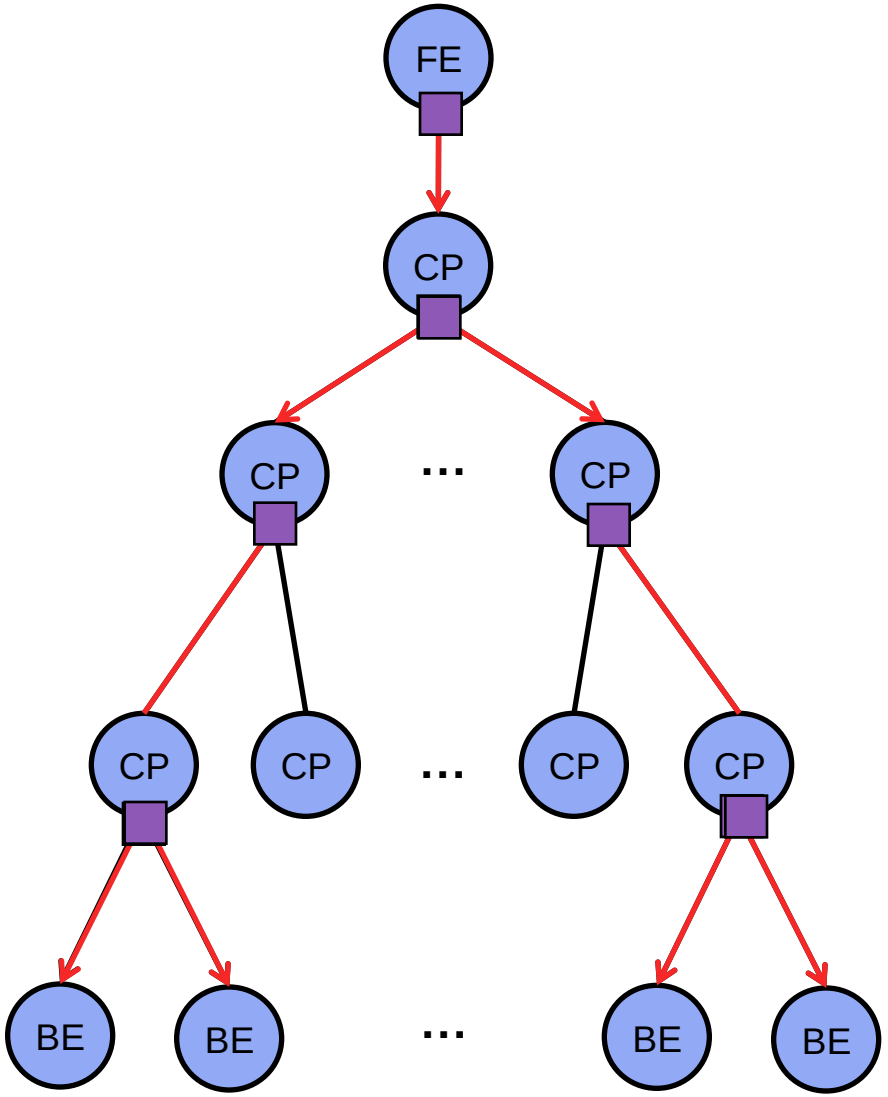
Logical channels called streams connect the front-end to the back-ends

Data is sent along a stream in a packet



# MRNet: An Easy-to-use TBŌN

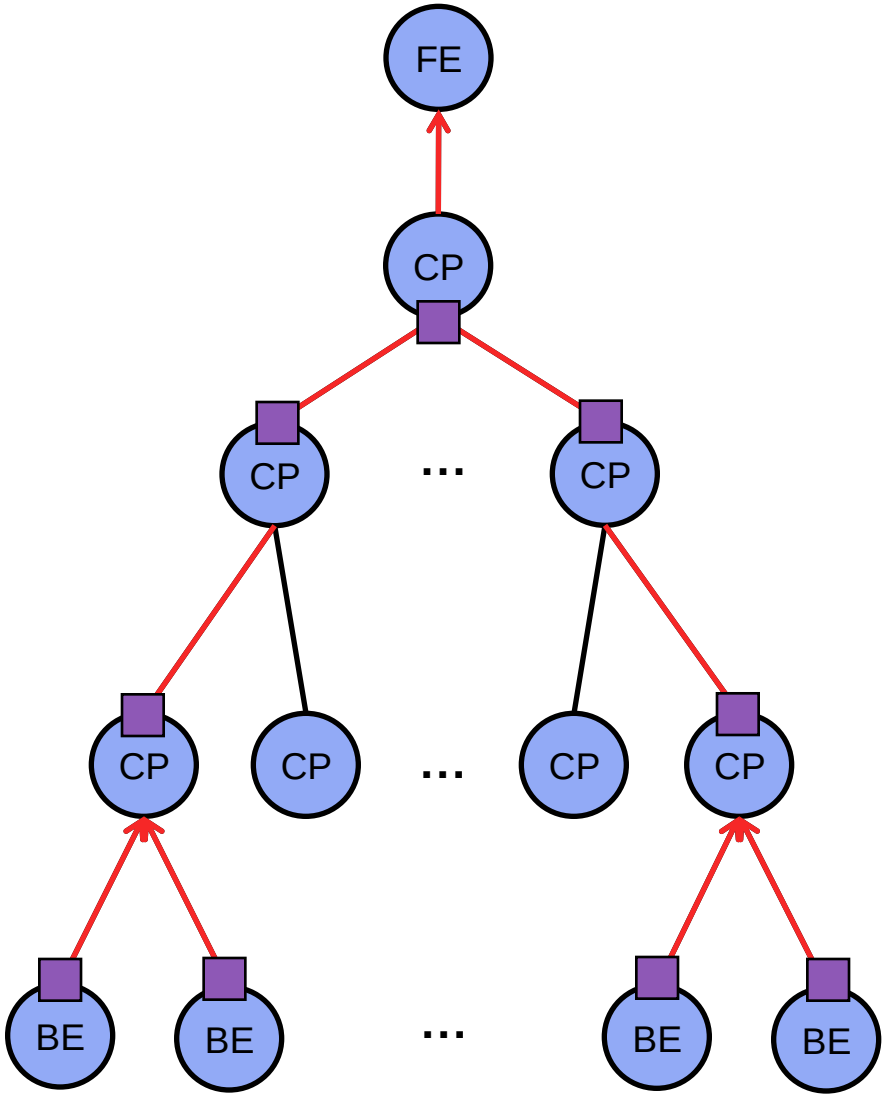
Easily multicast messages to backend nodes



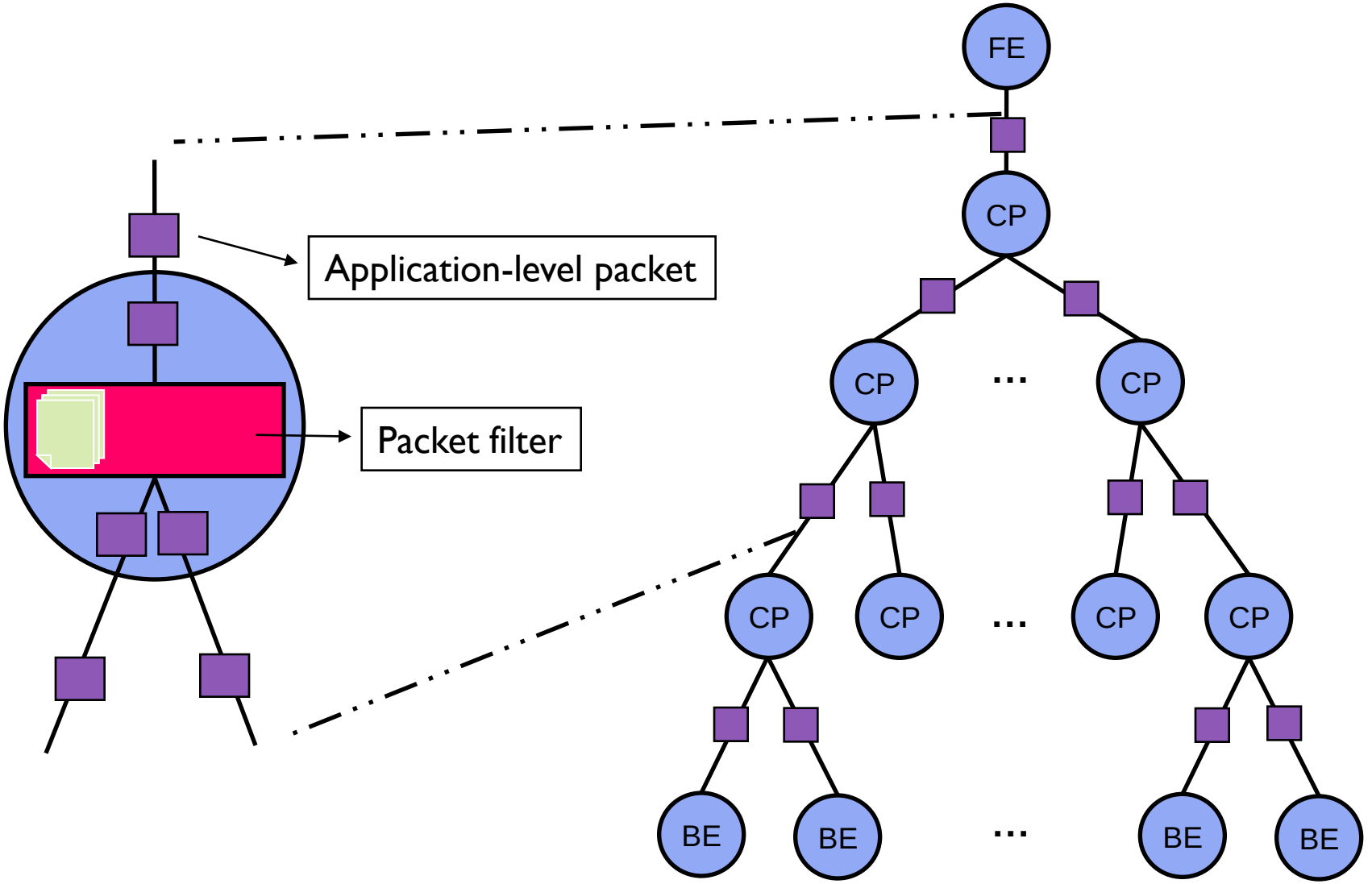
# MRNet: An Easy-to-use TBŌN

Easily multicast messages to backend nodes

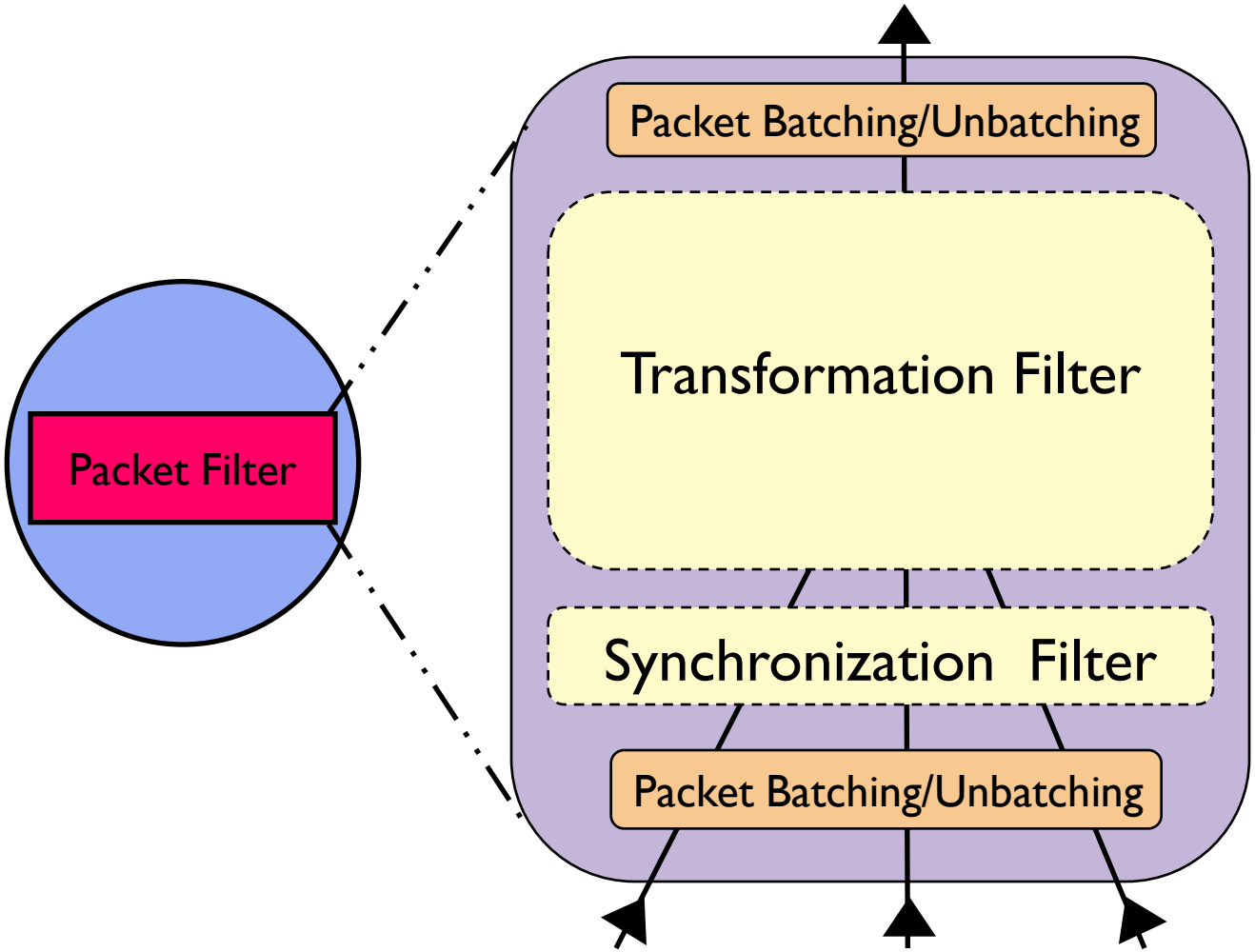
and aggregate data as it is sent to the frontend



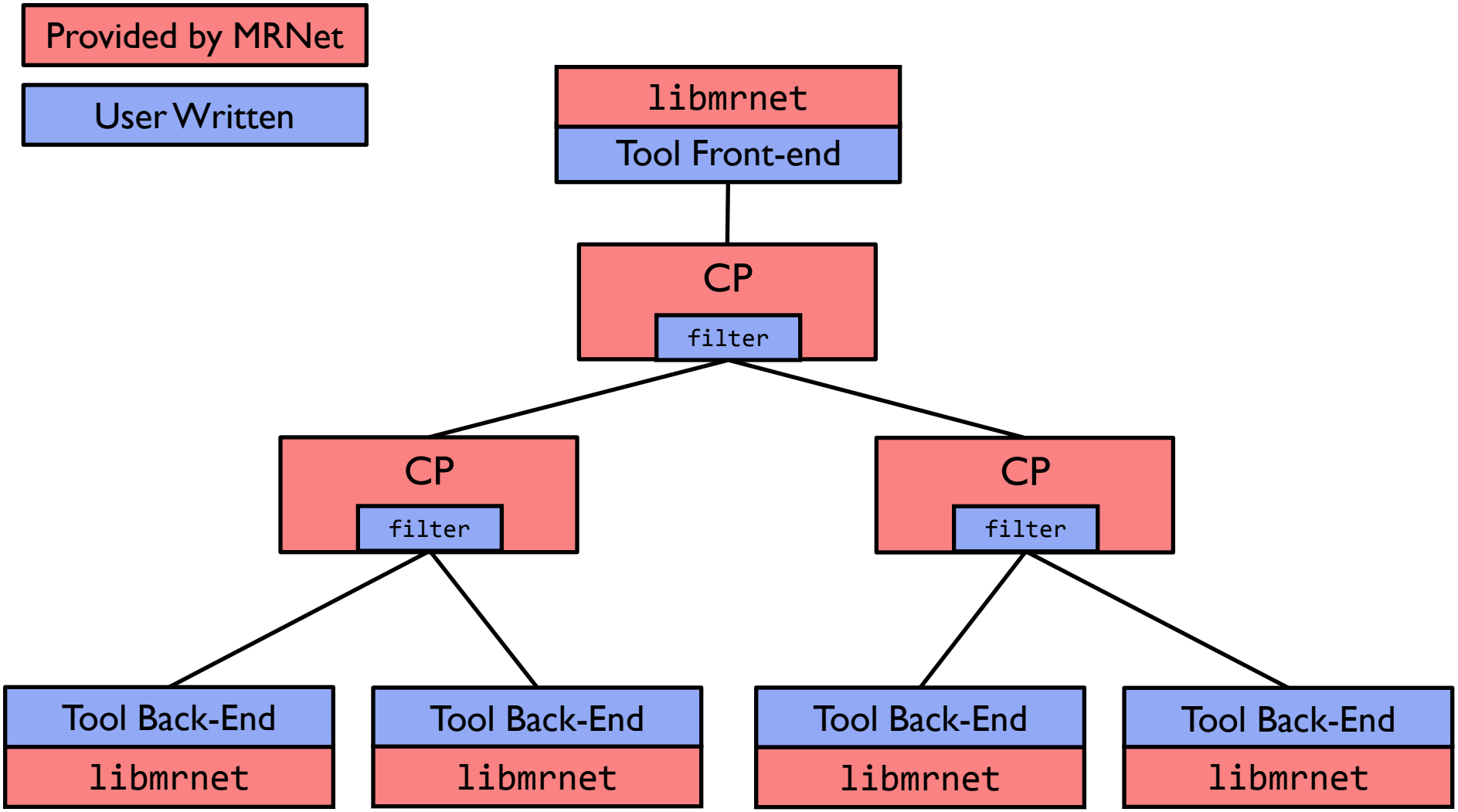
# MRNet: An Easy-to-use TBŌN



# MRNet Filters

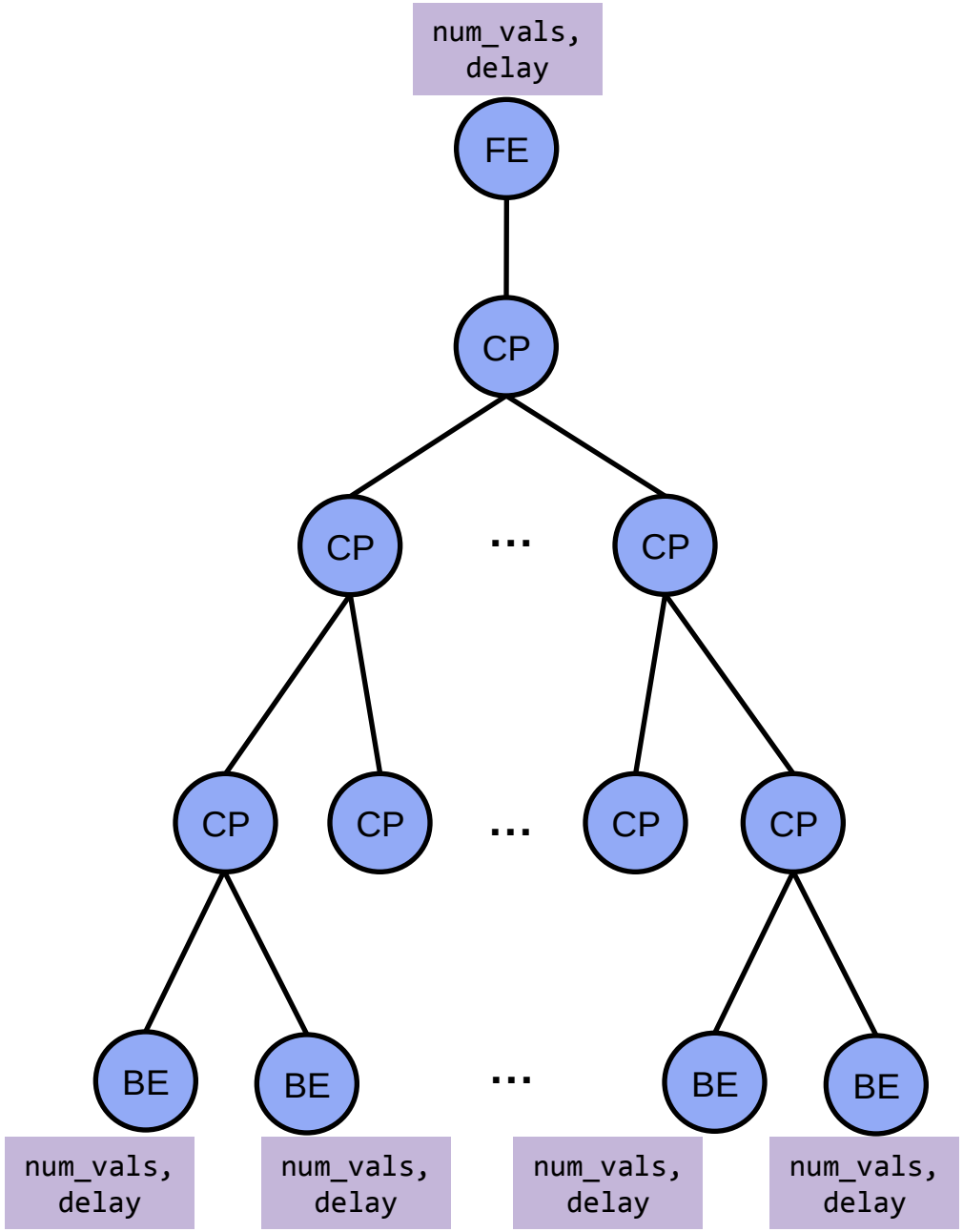


# MRNet Components



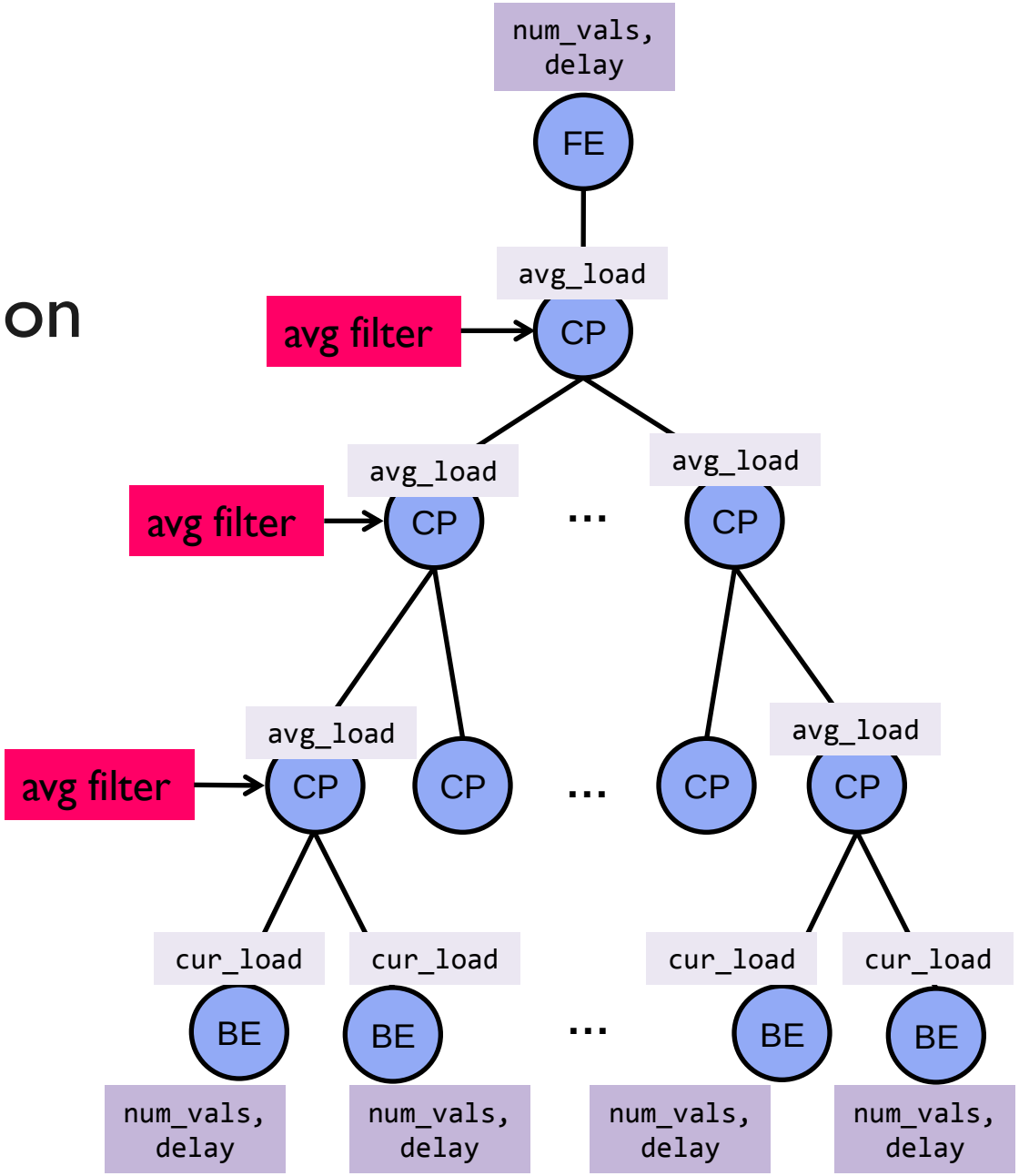
# Example MRNet Tool

Performance Tool:
gather load information
from backends



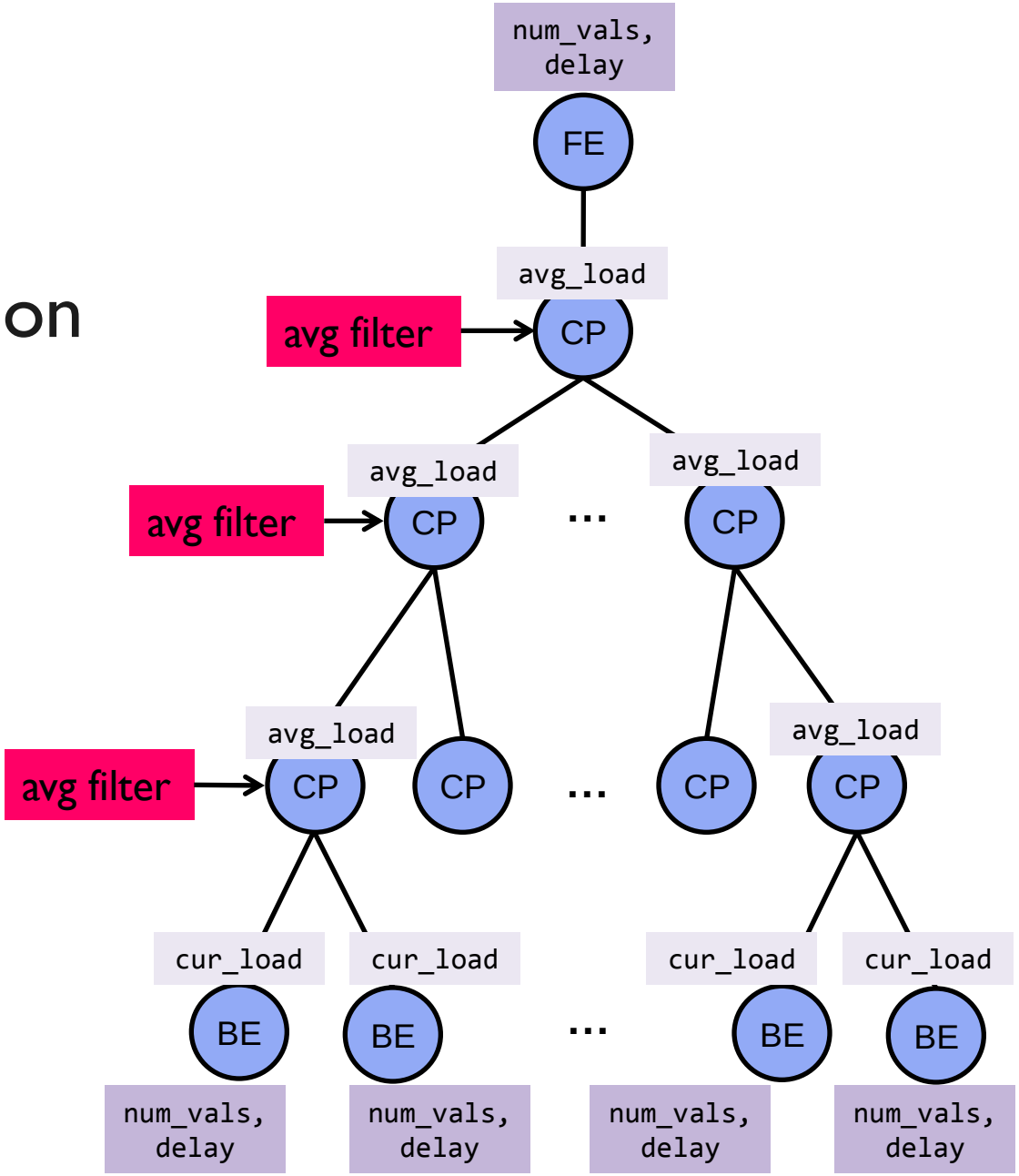
# Example MRNet Tool

Performance Tool:
gather load information
from backends



# Example MRNet Tool

Performance Tool:
gather load information
from backends



# Example Frontend Code

```
front_end_main(int argc, char ** argv) {

  Network * net = Network::CreateNetworkFE(topo_file, bckend_exe, &dummy_argv);
  int filter_id = net->load_FilterFunc(so_file, "LoadAvg");


  Communicator * comm = net->get_BroadcastCommunicator();
  Stream * strm = net->new_Stream(comm, filter_id, SFILTER_WAITFORALL);


  int tag = PROT_SUM;
  strm->send(tag, "%d %d", num_vals, delay);


  for (i = 0; i < num_vals; i++) {
    strm->recv(&tag, pkt);
    pkt->unpack("%d", &recv_val);
  }


  strm->send(PROT_EXIT, "");
  delete net;

}
```

# Example Frontend Code

```
front_end_main(int argc, char ** argv) {
  Network * net = Network::CreateNetworkFE(topo_file, bckend_exe, &dummy_argv);
  int filter_id = net->load_FilterFunc(so_file, "LoadAvg");

  Communicator * comm = net->get_BroadcastCommunicator();
  Stream * strm = net->new_Stream(comm, filter_id, SFILTER_WAITFORALL);

  int tag = PROT_SUM;
  strm->send(tag, "%d %d", num_vals, delay);

  for (i = 0; i < num_vals; i++) {
    strm->recv(&tag, pkt);
    pkt->unpack("%d", &recv_val);
  }

  strm->send(PROT_EXIT, "");
  delete net;
}
```

Create a new instance of a Network

# Example Frontend Code

```
front_end_main(int argc, char ** argv) {

  Network * net = Network::CreateNetworkFE(topo_file, bckend_exe, &dummy_argv);
  int filter_id = net->load_FilterFunc(so_file, "LoadAvg");

  Communicator * comm = net->get_BroadcastCommunicator();
  Stream * strm = net->new_Stream(comm, filter_id, SFILTER_WAITFORALL);

  int tag = PROT_SUM;
  strm->send(tag, "%d %d", num_vals, delay);

  for (i = 0; i < num_vals; i++) {
    strm->recv(&tag, pkt);
    pkt->unpack("%d", &recv_val);
  }

  strm->send(PROT_EXIT, "");
  delete net;

}
```

Get broadcast communicator and create a Stream that uses this communicator

# Example Frontend Code

```
front_end_main(int argc, char ** argv) {

  Network * net = Network::CreateNetworkFE(topo_file, bckend_exe, &dummy_argv);
  int filter_id = net->load_FilterFunc(so_file, "LoadAvg");


  Communicator * comm = net->get_BroadcastCommunicator();
  Stream * strm = net->new_Stream(comm, filter_id, SFILTER_WAITFORALL);


  int tag = PROT_SUM;
  strm->send(tag, "%d %d", num_vals, delay);


  for (i = 0; i < num_vals; i++) {
    strm->recv(&tag, pkt);
    pkt->unpack("%d", &recv_val);
  }


  strm->send(PROT_EXIT, "");
  delete net;

}
```

Send num_vals and delay to the BEs

# Example Frontend Code

```
front_end_main(int argc, char ** argv) {

  Network * net = Network::CreateNetworkFE(topo_file, bckend_exe, &dummy_argv);
  int filter_id = net->load_FilterFunc(so_file, "LoadAvg");


  Communicator * comm = net->get_BroadcastCommunicator();
  Stream * strm = net->new_Stream(comm, filter_id, SFILTER_WAITFORALL);


  int tag = PROT_SUM;
  strm->send(tag, "%d %d", num_vals, delay);


  for (i = 0; i < num_vals; i++) {
    strm->recv(&tag, pkt);
    pkt->unpack("%d", &recv_val);
  }


  strm->send(PROT_EXIT, "");
  delete net;

}
```

Receive and unpack Packet with a single int

# Example Frontend Code

```
front_end_main(int argc, char ** argv) {

  Network * net = Network::CreateNetworkFE(topo_file, bckend_exe, &dummy_argv);
  int filter_id = net->load_FilterFunc(so_file, "LoadAvg");


  Communicator * comm = net->get_BroadcastCommunicator();
  Stream * strm = net->new_Stream(comm, filter_id, SFILTER_WAITFORALL);


  int tag = PROT_SUM;
  strm->send(tag, "%d %d", num_vals, delay);


  for (i = 0; i < num_vals; i++) {
    strm->recv(&tag, pkt);
    pkt->unpack("%d", &recv_val);
  }


  strm->send(PROT_EXIT, "");
  delete net;

}
```

Teardown the Network

# Example Filter Code

```
const char * LoadAvg_format_string = "%d";


void LoadAvg(std::vector<PacketPtr> & pkts_in,
             std::vector<PacketPtr> & pkts_out,
             std::vector<PacketPtr> &, /* packets_out_reverse */
             void **,                  /* client data */
             PacketPtr &) {            /* params */


 int avg = 0;

 for (unsigned int i = 0; i < pkts_in.size(); i++) {
    PacketPtr cur_packet = pkts_in[i];
    int val;
    cur_packet->unpack("%d", &val);
    avg += val;
  }

  avg = avg / pkts_in.size();


  PacketPtr new_pkt (new Packet(pkts_in[0]->get_StreamId(),
                                pkts_in[0]->get_Tag(), "%d", avg));
  pkts_out.push_back(new_pkt);
}
```

# Example Filter Code

```
const char * LoadAvg_format_string = "%d";
```

Declare format of data expected by the filter

```
void LoadAvg(std::vector<PacketPtr> & pkts_in,
             std::vector<PacketPtr> & pkts_out,
             std::vector<PacketPtr> &, /* packets_out_reverse */
             void **,                  /* client data */
             PacketPtr &) {            /* params */


 int avg = 0;

 for (unsigned int i = 0; i < pkts_in.size(); i++) {
    PacketPtr cur_packet = pkts_in[i];
    int val;
    cur_packet->unpack("%d", &val);
    avg += val;
  }

  avg = avg / pkts_in.size();


  PacketPtr new_pkt (new Packet(pkts_in[0]->get_StreamId(),
                                pkts_in[0]->get_Tag(), "%d", avg));
  pkts_out.push_back(new_pkt);
}
```

# Example Filter Code

```
const char * LoadAvg_format_string = "%d";
```

Use generic function signature

```
void LoadAvg(std::vector<PacketPtr> & pkts_in,
             std::vector<PacketPtr> & pkts_out,
             std::vector<PacketPtr> &, /* packets_out_reverse */
             void **,                  /* client data */
             PacketPtr &) {            /* params */


 int avg = 0;

 for (unsigned int i = 0; i < pkts_in.size(); i++) {
    PacketPtr cur_packet = pkts_in[i];
    int val;
    cur_packet->unpack("%d", &val);
    avg += val;
  }

  avg = avg / pkts_in.size();


  PacketPtr new_pkt (new Packet(pkts_in[0]->get_StreamId(),
                                pkts_in[0]->get_Tag(), "%d", avg));
  pkts_out.push_back(new_pkt);
}
```

# Example Filter Code

```
const char * LoadAvg_format_string = "%d";


void LoadAvg(std::vector<PacketPtr> & pkts_in,
             std::vector<PacketPtr> & pkts_out,
             std::vector<PacketPtr> &, /* packets_out_reverse */
             void **,                  /* client data */
             PacketPtr &) {            /* params */

 int avg = 0;

 for (unsigned int i = 0; i < pkts_in.size(); i++) {
    PacketPtr cur_packet = pkts_in[i];
    int val;
    cur_packet->unpack("%d", &val);
    avg += val;
  }

  avg = avg / pkts_in.size();


  PacketPtr new_pkt (new Packet(pkts_in[0]->get_StreamId(),
                                pkts_in[0]->get_Tag(), "%d", avg));
  pkts_out.push_back(new_pkt);
}
```

Aggregate incoming packets

# Example Filter Code

```
const char * LoadAvg_format_string = "%d";


void LoadAvg(std::vector<PacketPtr> & pkts_in,
             std::vector<PacketPtr> & pkts_out,
             std::vector<PacketPtr> &, /* packets_out_reverse */
             void **,                  /* client data */
             PacketPtr &) {            /* params */


 int avg = 0;

 for (unsigned int i = 0; i < pkts_in.size(); i++) {
    PacketPtr cur_packet = pkts_in[i];
    int val;
    cur_packet->unpack("%d", &val);
    avg += val;
  }

  avg = avg / pkts_in.size();
```

Create new outgoing Packet

```
  PacketPtr new_pkt (new Packet(pkts_in[0]->get_StreamId(),
                                pkts_in[0]->get_Tag(), "%d", avg));
  pkts_out.push_back(new_pkt);
}
```

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);


  Stream * strm;
  PacketPtr pkt;


  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);


  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

# Example Backend Code

```
back_end_main(int argc, char ** argv) {
  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);

  Stream * strm;
  PacketPtr pkt;

  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);

  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

Create a new instance of a Network

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);

  Stream * strm;
  PacketPtr pkt;

  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);

  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

Declare some necessary variables

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);


  Stream * strm;
  PacketPtr pkt;

  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);


  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

Do an anonymous network receive

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);


  Stream * strm;
  PacketPtr pkt;


  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);


  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

Unpack Packet containing two ints

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);


  Stream * strm;
  PacketPtr pkt;


  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);


  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

Send current load up the stream, then sleep for specified time

# Using the Lightweight Backend API

- For those already using MRNet, learning to use the lightweight API is easy

```
int Stream::send(int tag,
                 char * format_string,
                 …);
```

```
int Stream_send(Stream_t * stream,
                int tag,
                char * format_string,
                …);
```

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);


  Stream *   strm;
  PacketPtr  pkt;


  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);


  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
    Network::CreateNetworkBE(argc,
                             argv);


  Stream *   strm;
  PacketPtr  pkt;


  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);


  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }

}
```

```
back_end_main(int argc, char ** argv) {

  Network_t * net =
    Network_CreateNetworkBE(argc,
                            argv);


  Stream_t * strm;
  Packet_t * pkt;


  Network_recv(net, &tag, pkt, &strm);
  Packet_unpack(pkt,"%d %d",&n,&delay);


  for (i = 0; i < n; i++) {
    Stream_send(strm,tag,"%d",
                get_load());
    sleep(delay);
  }

}
```

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network * net =
     Network::CreateNetworkBE(argc,
                              argv);

  Stream *   strm;
  PacketPtr  pkt;

  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);

  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

```
back_end_main(int argc, char ** argv) {

  Network_t * net =
     Network_CreateNetworkBE(argc,
                             argv);

  Stream_t * strm;
  Packet_t * pkt;

  Network_recv(net, &tag, pkt, &strm);
  Packet_unpack(pkt,"%d %d",&n,&delay);

  for (i = 0; i < n; i++) {
    Stream_send(strm,tag,"%d",
                get_load());
    sleep(delay);
  }
}
```

# Example Backend Code

```
back_end_main(int argc, char ** argv) {

  Network *  net =
     Network::CreateNetworkBE(argc,
                              argv);

  Stream *   strm;
  PacketPtr  pkt;

  net->recv(&tag, pkt, &strm);
  pkt->unpack("%d %d",&n,&delay);

  for (i = 0; i < n; i++) {
    strm->send(tag, "%d",
               get_load());
    sleep(delay);
  }
}
```

```
back_end_main(int argc, char ** argv) {

  Network_t *  net =
     Network_CreateNetworkBE(argc,
                             argv);

  Stream_t *  strm;
  Packet_t *  pkt;

  Network_recv(net, &tag, pkt, &strm);
  Packet_unpack(pkt,"%d %d",&n,&delay);

  for (i = 0; i < n; i++) {
    Stream_send(strmtag, "%d",
                get_load());
    sleep(delay);
  }
}
```

# New Lightweight Backend Library

- Can embed C library in application processes
- Lightweight MRNet BE can run on reduced node kernels (such as BlueGene compute node)
- Avoids the need to deal with threading in application or tool

# Lightweight MRNet

- Standard MRNet back-end as part of tool
  - C++ library
  - Multi-threaded
  - Dedicated thread receives data
  - Filtering at back-ends
- Lightweight back-end as part of C-based tool or application
  - C library
  - Single-threaded
  - No filtering at back-end

# Some MRNet Users

- Stack Trace Analysis Tool (STAT, LLNL)
- Cray Application Termination Processing (ATP, Cray)
- TotalView using TBON-FS (TotalView and Univ. Wisconsin)
- TAU over MRNet performance tool (ToM, Univ. Oregon)
- Open|SpeedShop, Component Based Tool Framework (CBTF, Krell Institute)
- Paradyn Performance Tool (Univ. Wisconsin)
- Group File Operations & TBON-FS (Univ. Wisconsin)
- Clustering algorithms (Mean shift, Univ. Wisconsin Vision Group)
- On-line detection of large scale application structure (BSC)
- …

# Conclusions

- MRNet provides infrastructure for scalable communication and computation
  - Runs full-scale on large systems, for example:
    - Jaguar – 216K processes
    - BlueGene/L – 208K processes
  - Used by a wide variety of tools
  - Handles the hard work of tool scaling
  - Provides fault tolerance support
- MRNet can be integrated with tools written in either C or C++

# Questions?

MRNet 3.0 Available Soon!

http://www.paradyn.org/html/downloads.html

Manuals and Publications Available:

http://www.paradyn.org/mrnet

Additional Questions?

jacobson@cs.wisc.edu