«هو العلیم»

موضوع سمینار:

مکانیزمهای کنترل ازدحام در

TCP

و

مروری بر عملکرد TCP در

Wireless Sensor Networks

استاد راهنما:

آقای دکتر یغمایی

دانشجو:

الهه بلندی

تابستان ۸۶

فهرست مطالب

فصل دوم

۱-۱)تعاریف (definitions):

سگمنت (Segment): به بسته های TCP (Data,Ack) اصطلاحا سگمنت گفته می شود.

SMSS(Sender Maximum Segment Size): اندازه بزرگترین سگمنتی که فرستنده می تواند ارسال کند. این مقدار براساس حداکثر واحد انتقال در شبکه ، الگوریتمهای تعیین MTU ، RMSS یا فاکتورهای دیگر تعیین می شود. این اندازه شامل هدر بسته و option نمی باشد.

RMSS(Receiver Maximum Segment Size): سایز بزرگترین سگمنتی که گیرنده می تواند دریافت کند. که در یک ارتباط در فیلد MSS در option توسط گیرنده تعیین می شود و شامل هدر و option نمی باشد.

rwnd(Receiver Window) : طول پنجره سمت گیرنده.

cwnd(Congestion Window): نشان دهنده وضعیت متغیر TCP است که میزان داده در شبکه را محدود می کند.در هر لحظه , حجم داده در شبکه به اندازه مینیمم cwnd و rwnd می باشد.

۱-۲)مکانیزمهای کنترل ازدحام در شبکه TCP:

در یک شبکه زمانیکه ترافیک بار از ظرفیت شبکه بیشتر می شود ، ازدحام اتفاق می افتد.که به منظورکنترل ازدحام در شبکه الگوریتمهای متفاوتی وجود دارد. در یک ارتباط

، لایه شبکه تا حدی قادر به کنترل ازدحام در شبکه است اما راه حل واقعی برای اجتناب از ازدحام پایین آوردن نرخ تزریق داده در شبکه است. TCP با تغییر سایز پنجره ارسال تلاش میکند که نرخ تزریق داده را کنترل کند.

شناسایی ازدحام اولین گام در جهت کنترل آن است.

در گذشته ،شناسایی ازدحام به راحتی امکانپذیر نبود. از نشانه های آن وقوع Timeout بدلیل اتلاف بسته یا وجود noise در خط ارتباطی یا اتلاف بسته ها در روترهای پر ازدحام و ... را می توان نام برد. اما امروزه از آنجا که اکثرا تکنولوژی بستر ارتباطی از نوع فیبر می باشد اتلاف بسته ها که منجر به خطای ارتباطی شود بندرت اتفاق می افتد. و از طرفی وقوع Timeout در اینترنت بدلیل ازدحام می باشد.

در همه الگوریتمهای TCP فرض بر این است که وقوع Timeout بدلیل ازدحام شبکه است.

در شروع یک ارتباط در شبکه ،سایز پنجره مناسب تعیین می شود. گیرنده بر اساس سایز بافر خود می تواند سایز پنجره را تعیین کند. اگر میزان داده های ارسالی از فرستنده در حد سایز پنجره باشد ،مشکلی پیش نمی آید. در غیر اینصورت در سمت گیرنده Overflow اتفاق می افتد. پس بطور کلی با دو مسئله مواجه هستیم:

۱- ظرفیت شبکه

۲- ظرفیت گیرنده

که دریک ارتباط باید این دو مورد را در نظر گرفت.

فرستنده در هنگام ارسال ،سایز دو پنجره را در نظر می گیرد.

۱- پنجره سمت گیرنده

۲- پنجرا ازدحام

که سایز پنجره ارسال به اندازه مینیمم این دو مقدار تعیین می شود.

کنترل ازدحام در شبکه TCP از بالا رفتن ظرفیت شبکه جلوگیری می کند . در واقع به فرستنده اجازه مس دهد نرخ ارسال داده در شبکه را به منظور جلوگیری از ازدحام تنظیم نماید.

مکانیزمهای کنترل ازدحام که توسط TCP حمایت می شوند عمدتا شامل ۴ مرحله اصلی می باشند که عبارتند از:

۱- Slow Start

۲- Congestion Avoidance

۳- Fast Retransmission

۴- Fast Recovery

که در ادامه هر یک به تفصیل شرح داده می شوند.

۱-۲-۱)Slow Start: در شروع یک ارتباط ،ارسال داده ها به سمت گیرنده به اندازه حداکثر ظرفیت سمت گیرنده انجام نمی شود. بلکه فرستنده تعدادی بسته در شروع ارسال می کند و منتظر دریافت Ack بسته های ارسالی می شود. و سپس بتدریج نرخ

ارسال را افزایش می دهد . این مکانیزم به فرستنده TCP این اجازه را می دهد که وضعیت موجود شبکه اعم از پهنای باند در دسترس را شناسایی کند.

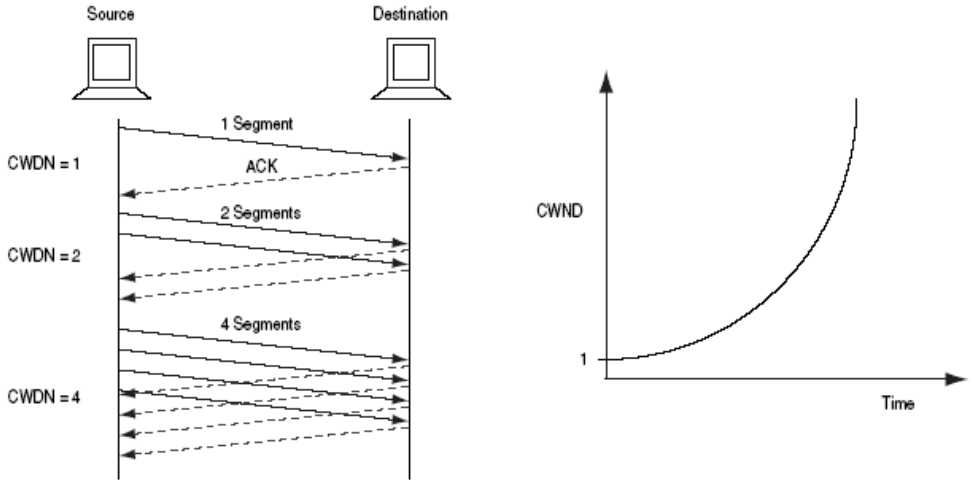
مکانیزم Slow Start درموارد زیر کاربرد دارد:

۱- در شروع هر ارتباط مبتی بر TCP.

۲- در شروع مجدد یک ارتباط TCP بعد از مدتی بیکاری.

۳- در شروع مجدد یک ارتباط TCP بعد از وقوعTimeout .

شکل(۱) مکانیزم Slow Start در TCP را به تصویر کشیده است. در مرحله Slow Start فرستنده باید سایز پنجره ازدحام (Cwnd که میزان ترافیک در شبکه را مشخص می کند) داشته باشد. تعداد سگمنتهای ارسالی از سمت فرستنده در هر لحظه به اندازه مینیمم سایز پنجره ازدحام و سایز پنجره سمت گیرنده می باشد.



شکل(۱)

در شروع یک ارتباط سایز این پنجره یک سگمنت است. فرستنده TCP با ارسال یک سگمنت وارد فاز Slow Start شده و منتظر دریافتAck می شود. بعد از دریافت Ack فرستنده سایز پنجره ارسال را از یک به دو افزایش می دهد و دو سگمنت فرستاده می شود.زمانیکه Ack این دو سگمنت از سمت گیرنده دریافت شد ، فرستنده سایز پنجره ارسال را از ۲ به ۴ افزایش می دهد و ۴ سگمنت فرستاده می شود. رشد نمایی سایز پنجره ارسال (Cwnd) ادامه می یابد تا زمانیکه به حداکثر سایز پنجره سمت گیرنده برسد یا بسته ها بدلیل ازدحام در شبکه اتلاف شوند. در فاز بعدی (Congestion Avoidance) چگونگی برخورد TCP با اتلاف بسته ها شرح داده میشود.

فرستنده TCP به دو طریق می تواند اتلاف بسته ها در شبکه را تشخیص دهد:

۱- دریافت ack تکراری.

۲- سر ریز شدن تایمر.

عدم وجود یک سگمنت در بین سگمنتهای ارسالی در یک پنجره باعث ایجاد Ack تکراری در مقصد می شود. (یادآوری می شود که TCP , Ack منفی یا Ack حاوی شماره بسته نمی فرستد.)

برای مثال , زمانیکه مقصد داده را تا بایت ۲۰۰۰ دریافت می کند در پاسخ Ack داده ۲۰۰۱ را می فرستد به این مفهوم که سگمنت بعدی که گیرنده انتظار دریافت آنرا دارد از

بایت ۲۰۰۱ شروع می شود . اگر یک سگمنت در روترهای میـانی تلـف شـود , در مقصـد بسته های بعدی به ترتیب رسیدن بافر می شوند و از آنجاییکه گیرنـده سـگمنت بعـدی مورد انتظـار را دریافـت نکـرده بـه فرسـتادن Ack سـگمنت ۲۰۰۱ ادامـه مـی دهـد. از آنجاییکه دریافت Ack تکراری همچنین دلالت بر عدم دریافت سگمنتها به ترتیـب دارد, در نتیجه مبدا TCP از سه Ack تکراری به نشانه اتلاف بسته ها استفاده می کند. اتلاف آخرین بسته در یک پنجره ارسالی از سگمنتها باعث سرریز شدن تایمر در مبدا می شود , زیرا سگمنت بعدی جهت تولید Ack وجود ندارد.

فرستنده TCP انتظار دریافت Ack از سمت مقصد به نشانه دریافت موفق بایتهای جدید در رشته داده ها را دارد. هر زمانی که فرستنده یک سگمنت را ارسال می کند یک تـایمر شروع بکار می کند و منتظر دریافت Ack می شود. اگر تـایمر قبـل از اینکـه Ack داده های ارسالی دریافت شود, سرریز شود فرستنده TCP فـرض را بـر ایـن مـی گـذارد کـه سگمنت گم شده یا آسیب دیده است . در نتیجه مجددا آنرا ارسال می کند.

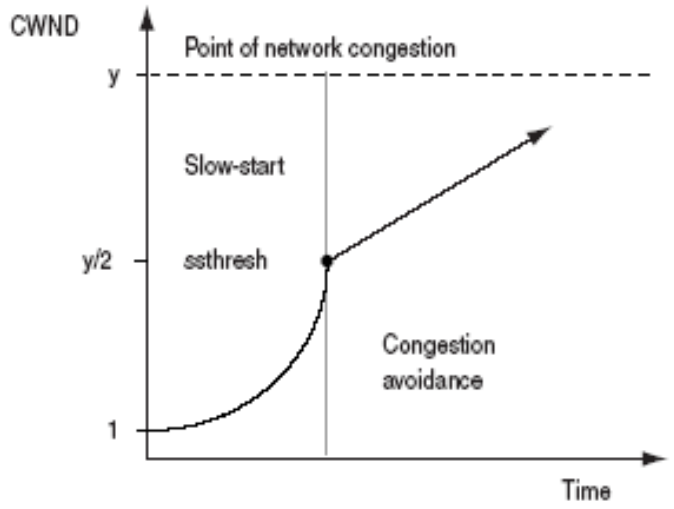
۲-۲-۱)Congestion Avoidance:

زمانیکه فرستنده TCP متوجه اتلاف بسته هـا در شـبکه شـد متغیـر ssthreshیـا حـد آستانه ناحیه slow start (slow-start threshold)را به اندازه نصف cwnd جـاری

تنظیم می کند. در واقع فرستنده نرخ ارسال خود را با برگشت به فاز Slow Start کاهش می دهد. با این تفاوت که افزایش نمایی نرخ ارسال تا رسیدن به سایز پنجره ارسال به حد آستانه (ssthresh) ادامه می یابد. در این نقطه ، فرستنده سایز پنجره ارسال را (cwnd) بطور خطی افزایش می دهد (با یک سگمنت در هر RTT).

در واقع به آهستگی نرخ ارسال را افزایش می دهد تا به مقدار cwnd قبلی که در آن نقطه اتلاف بسته رخ داده بود برسد.

زمانیکه cwnd مساوی یا کمتر از حد آستانه ssthresh است فرستنده TCP در فاز slow-start است. و زمانیکه cwnd بزرگتر از ssthresh است فرستنده TCP در فاز Congestion Avoidance قرار دارد. که در شکل (۲) به تصویر کشیده شده است.



شـــکل

(۲)

فاز slow-start و Congestion Avoidance سبب می شود TCP بـا وقـوع اتـلاف بسته ها سایز پنجره ارسال را به نصف کاهش دهد . اگر وجود ازدحام در شـبکه بـه طـور مرتب باعث اتلاف بسته ها شود میزان ترافیک وارد شده به شـبکه و نـرخ ارسـال توسـط فرستنده TCP بطورنمایی کاهش می یابد. که این امر به روترهای میانی اجازه مـی دهـد تا صفهای خود را خالی کنند.

۳-۲-۱)Fast Retransmission:

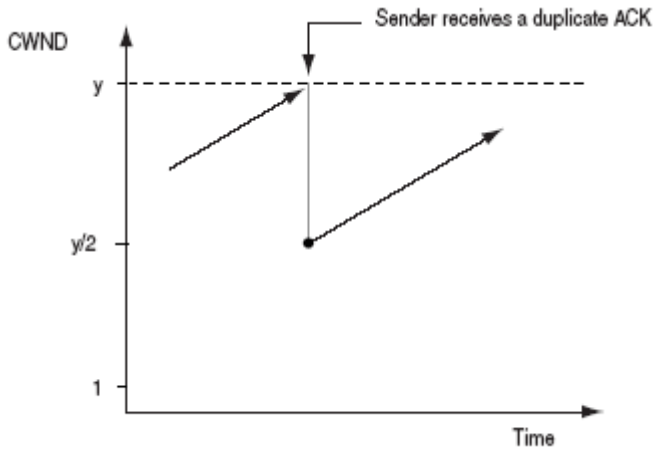
همانطور که قبلا گفته شد ، TCP با دریافت Ack تکراری متوجه اتلاف بسته ها میشود. از طرفی دریافت Ack تکراری همچنین می تواند به مفهوم دریافت بسـته هـا بـر خـلاف ترتیب اصلی باشد. پس بجای نشان عکس العمل سریع نسبت به دریافت Ack تکراری بـا ارسال مجدد بسته گم شده، فرستنده TCP منتظر میشود تا ۳ Ack تکـراری دریافـت کند.

Fast Retransmission کارایی TCP را به ۳ طریق افزایش دهد:

- اگر بسته ها به ترتیب دریافت نشده باشند از ارسال مجدد آنها و در نتیجه پر کردن ظرفیت شبکه جلوگیری می کند.
- بهره وری بیشتر و بازدهی بالا خطوط ارتباطی را موجب می شود.
- در صورت اتلاف بسته ها ، TCP قبل از سرریز تایمر اقدام به ارسال مجدد بسته می کند. و منتظر سرریز شدن تایمر نمی شود.

۴-۲-۱)Fast Recovery:

زمانیکه مبدا TCP ، Ack تکراری دریافت می کند ،داده ها به سمت مقصد در جریان هستند زیرا در مقصد اگر سگمنتها به ترتیب دریافت نشوند Ack تکراری تولید می شود. در این مورد فرستنده TCP به طور ناگهانی نرخ ارسال داده را با رفتن به فاز slow-start کاهش نمی دهد. در عوض بعد از ارسال مجدد سگمنت گم شده (در پاسخ به ۳ Ack تکراری) مبدا TCP میزان cwnd را به نصف کاهش می دهد و وارد فاز Congestion Avoidanceمی شود. این عملیات باعث بازدهی بهتر TCP می شود. که این عملیات در شکل (۳) به تصویر کشیده شده است.

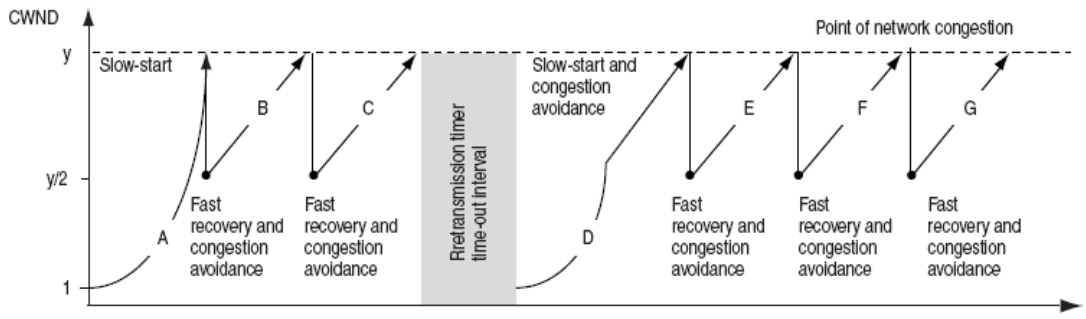


شکل (۳)

Fast Recovery از خالی شدن جریان TCP بین مبدا و مقصد بعد از اتلاف یک بسته و ارسال مجدد آن جلوگیری می کند. این فاز کارایی TCP را با عدم نیاز به برگشت به فاز slow-start و افزایش تدریجی ظرفیت شبکه بعد از اتلاف یک بسته ایجاد می شود. در مقابل, در حالیکه فاز Fast Recovery کارایی TCP را زمانیکه یک بسته در یک پنجره ارسالی گم می شود افزایش میدهد.کارایی را زمانیکه چندین بسته در یک پنجره اتلاف می شوند افزایش نمی دهند.

مثال:

شکل ۴ بازدهی جریان TCP را نمایش می دهد.



شکل (۴)

در شروع یک ارتباط TCP ،مبدا اتصال سعی در اجتناب از بالا بردن ترافیک شبکه دارد. همانطور که در منحنی A دیده می شود ارتباط با فاز slow-start آغاز می شود و سرعت نرخ ارسال افزایش می یابد تا ظرفیت جاری تشخیص داده شود. فاز slow-start تا زمانیکه Ack تکراری دریافت شود ادامه می یابد. همانطور که در منحنی قسمت B نشان داده شده مبدا TCP میزان حد آستانه ssthresh را به نصف میزان cwndجاری تنظیم می کند و Fast Restartانجام می شود و به فاز Congestion Avoidance می رسد تا دوباره به میزان cwnd قبلی که منجر به اتلاف بسته شده بود برسد.

بعد از شروع مجدد با فاز Congestion Avoidance نرخ ارسال افزایش می یابد تا زمانیکه Ack تکراری دریافت شود . همانطور که در منحنی C نشان داده شده است،

فرستنده میزان ssthresh را مساوی نصف cwnd جاری تنظیم می کند و بازیابی مجدد همراه با Congestion Avoidance انجام میشود.

حال اگر آخرین سگمنت در پنجره ارسال در یکی از روترهای میانی آسیب ببیند ،در اینصورت مبدا TCP منتظر سرریز شدن تایمر است قبل از اینکه آخرین سگمنت را مجددا ارسال کند. همانطور که در منحنی D دیده می شود ، مبدا TCP میزان ssthresh را به نصف مقدار cwnd جاری تنظیم می کند و مجددا به فاز-slow startبر میگرددو با ازدحام برخورد می کند.

۳-۱)پیشرفتهای جدید در زمینه کنترل ازدحام در TCP:

در طی سالها ،پیاده سازی TCP منجر به طراحی الگوریتمهای متفاوتی به منظور کنترل ازدحام در شبکه و ایجاد یک ارتباط مناسب گشته است. در این زمینه الگوریتمهایی مثل Tahoe، Reno، New Reno، و Vegas مطرح هستند.که در ادامه نقاط ضعف و قوت هر یک را بیان می کنیم.

۱-۳-۱)TCP Tahoe:

اولین پیاده سازی از این نسخه TCP در سال ۱۹۸۸ در سیستم BSd۴,۳ یونیکس بود. و این نسخه پیاده سازی اصلی مکانیزم پیشنهادی توسط Van Jacobson بود. Tahoe شامل ۳ مرحله اساسی است.

Slow-Start، Congestion Avoidance، Fast Retransmission.

در طول Slow-Start ، در شروع یک ارتباط (cwnd=1) یک بسته ارسال مـی شـود. و سایز پنجره ارسال با دریافت Ack از سمت گیرنده بـه طـور نمـایی افـزایش مـی یابـد و ناحیـهSlow-Start ادامـه مـی یابـد تـا زمانیکـه cwnd <ssthresh . در طـول فـاز Congestion Avoidance

(cwnd= ssthresh) اتلاف بسته نشان دهنده ازدحام در شبکه است.

Tahoe در چنین مواقعی به شرح ذیل عمل می نماید.

۱- اگر در هنگامیکه اتلاف رخ می دهدcwnd = wباشد سایز پنجره ارسال بین 0.5w تا w بسته تنظیم میشود با این فرض که شبکه می تواند این تعداد بسته را هدایت کند.

۲- ssthresh را به اندازه0.5w تنظیم می کند و TCP بـه فـاز Slow-Start بـر مـی گردد یعنی جائیکه cwnd=۱

۳- با دریافت یک Ack جدید در حالیکه cwnd >sshresh ،سایز پنجره ارسـال یعنـی cwnd طبق رابطه زیر افزایش می یابد.

Cwnd+=1/ cwnd

فاز Fast Retransmission درواقع Tahoe را از وقوع Time out بدلیل اتلاف بسته ها حمایت می کند. دریافت Ack تکراری، دلالت بر اتـلاف بسـته هـا دارد و Tahoe در این شرایط به فاز Slow-Start بر می گردد. و با دریافت چندین Ack تکراری ،شروع به ارسال مجدد بسته ها می کند و منتظر وقوعTime out (سرریز شدن تایمر) نمی شود.

۲-۳-۱) TCP Reno:

الگوریتم Reno در سال ۱۹۹۵ مطرح شد و شامل تمام مکانیزمهای مطرح شده در الگوریتم Tahoe است. و علاوه بر آنها در Reno فازCongestion Avoidance با مکانیزم Fast Recovery توسعه می یابد. همانند Tahoe ،فاز Congestion Avoidance زمانیکه cwnd= ssthresh است شروع می شود. در این زمان cwnd با دریافت هر Ack فقط به اندازه 1/ cwnd افزایش می یابد. یعنی بصورت خطی رشد می کند.

و بطور مشابه فاز Fast Retransmission با دریافت سه Ack تکراری آغاز می شود. آنچه که باعث اختلاف بین Tahoe وReno است در واقع فاز Fast Recovery است. زمانیکه سومین Ack تکراری دریافت می شود ، ssthresh=0.5*cwnd و cwnd= ssthresh+3تنظیم میکنیم.

(سه Ack ،که هر Ack تکراری دلالت بر این دارد که گیرنده از دریافت سه بسته گم شده صرفنظر کرده است.)

سپس Reno میزان cwnd را تغییر می دهد تا زمانیکه Ack مناسب از بسته هایی که مجددا ارسال کرده را دریافت کند. در اینحال مجدداcwnd= ssthresh تنظیم می گردد.

بطور خلاصه, هر Ack تکراری فرستنده TCP را از اتلاف یک بسته آگاه می سازد و مکانیزم Fast Recovery بمنظور بازیابی اتلاف بسته ها میزان cwnd را تغییر می دهد. زمانیکه بسته گم شده احیا شد (در گیرنده دریافت شد) مجددا cwnd = cwnd /2 .درنتیجه Reno با استفاده از Fast Recovery بطور پیوسته به فاز Congestion Avoidance انتقال می یابد.

یکی از نقاط ضعف الگوریتم Reno نحوه برخورد آن با اتلاف چندین بسته در یک پنجره ارسال می باشد. Reno رویه Fast Recovery را چندین بار اجرا میکند تا با اتلاف چندین بسته در شبکه مقابله نماید و مرتبا مقادیر cwnd و ssthresh را تغییر میدهد. در پایان ,این امر در مواقعی منجر به وقوع Time Out می شود که منجر به تغییر cwnd به اندازه یک بسته می شود و کارایی را تحت تاثیر قرار میدهد. علاوه بر این , Reno ترافیک انفجاری در شبکه را نمی تواند به خوبی تحمل کند, که این امر به نحوه ارسال Ack بر می گرددکه در هر RTT فقط اتلاف یک بسته اعلام می شود.

۳-۳-۱)TCP New Reno:

New Reno بطور هوشمندانه نقاط ضعف Reno را بهبود می بخشد , بخصوص فازهای Slow Start و Fast Recovery را. در واقع بطور تدریجی سایز پنجره را تغییر می دهد و علاوه بر این اتلاف چندین بسته در یک پنجره را شناسایی می کند. در ابتدا لازم است مفهوم Partial Ack درک شود. یک Partial Ack , Ack مربوط به تعدادی از بسته ها و نه همه آنهاست, که در ابتدای ناحیه Fast Recovery متمایز می شوند.

Partial Ack سبب تغییر حالت New Reno به فـاز Fast Recovery نمـی شـود بلکه Partial Ack دریافتی در طول Fast Recovery دلالت بر این دارد که بسته ای که بلا فاصله بعد از بسته ای که Ack آن آمـده ,گـم شـده و بایـد مجـددا ارسـال شـود. بنابراین زمانیکه چندین بسته گم می شوند , New Reno می تواند بدون وقوع Time Out اتلاف را بهبود بخشد (بازیابی کند). در واقع عملکرد New Reno زمانی برجسـته و متمایز میشود که چندین بسته در یک پنجـره ارسـال گـم شـوند. در طـول فـاز Fast Recovery استفاده از Partial Ack بطور قابل تـوجهی میـزان وقـوع Time Out را کاهش میدهد.

۴-۳-۱)TCP Vegas:

Vegas نسـبت بـه الگـوریتم Reno دو فـاز Congestion Avoidance و Fast Retrasmission را بهبود می بخشـد. در واقـع بـه TCP توانـایی شناسـایی ازدحـام و تنظیم نرخ ارسال را بر اساس آن می دهد و مکانیزم ارسال مجدد بهینه همچنـین منجـر به این می شود که در مورد بسته های اتلاف شده , به منظور زمان ارسال تصـمیم گیـری بهتری انجام شود.

در طول ناحیـه Slow Start در الگـوریتم Vegas , cwnd بطـور نمـایی افـزایش مـی یابد.و در طول هر RTT بدون تغییر باقی می ماند.این امر مقایسه بین نرخ ارسـال مـورد انتظار و نرخ ارسال واقعی را محقق می سازد. بنابراین cwnd بطـور مناسـب تنظـیم مـی

شود. اگر نرخ واقعی کمتر از نرخ مورد انتظار باشد ، سپس Vegas به فاز Congestion Avoidance وارد می شود.

Vegas سه متغیر را در طول فاز Congestion Avoidance ذخیره می کند . α به عنوان حد آستانه که بوسیله مدیر شبکه انتخاب می شود . β یک متغیر حد آستانه پایین و Δ اختلاف بین نرخ ارسال مورد انتظار و واقعی. در طول مقایسه, cwnd بطور خطی افزایش می یابد اگر $\Delta < \alpha$ و کاهش می یابد بصورت خطی اگر $\Delta > \beta$ باشد. هدف Vegas در واقع تداوم ارسال است تا زمانیکه Δ در بازه (α، β) باشد. در طول روال Fast Retransmission تغییر یافته , Vegas به طرق زیر به منظور شناسایی سریعتر اتلاف عمل می کند:

۱- اختلاف بین زمان جاری و برچسب زمانی تعریف شده را چک می کند که اگر بزرگتر از Time Out باشد در مقابل رسید اولین Ack تکراری , بسته مجددا ارسال می شود , بدون انتظار برای دومین Ack تکراری.

۲- زمانیکه اولین یا دومین Ack دریافت شد (به عبارت دیگر , نه Ack تکراری) بعد از انتقال, مجددا چک می شود که اگر Time Out در حال افزایش باشد در اینحال ارسال مجدد سریعا شروع می شود.

۳- Cwnd در یک حالت کاهش می یابد. بسته ای که ارسال مجدد شده است قبلا بعد از آخرین کاهش cwnd . فرستاده شده باشد ، cwnd کاهش می یابد فقط به منظور اجتناب از ازدحام پر خطر.

پس از بررسی شرایط ازدحام و مکانیزمهای کنتـرل ازدحـام در ارتبـاط TCP ، در ادامـه عملکرد TCP را در شبکه های حسی مورد بررسی قرار می دهیم.

۱-۲)عملکرد بهینه TCP در شبکه های بی سیم حسی:

برنامه های کاربردی زیادی در شبکه های بی سیم حسی نیاز به برقراری اتصال بـا شـبکه های خارجی دارند تا بتوانند ارتباط موجودیتها را با سنسورها کنترل نمایند. با اسـتفاده از پروتکل TCP/IP در شبکه حسی, اتصالات خارجی به راحتی فراهم می شود.

در یک شبکه حسی مبتنی بر پروتکل TCP/IP, TCP می تواند به منظـور مـدیریت از راه دور و برنامه ریزی مجدد سنسورها استفاده شود.

هر چند که نرخ خطای بـالا در شـبکه هـای حسـی بـا چنـدین نـود (پـرش) منجـر بـه ناکارآمدی انرزی و در نتیجه طول عمر پایین شبکه های حسی می شـود, در ایـن جـا دو دیدگاه متفاوت به منظور عملکرد بهینه انـرزی از پروتکـل TCP در شـبکه هـای حسـی مطرح و بررسی می شود:

(DTC) Distributed TCP Cachimg

(TSS)TCP Support for Sensor networks

هر دو مکانیزم به سنسورهای نودهای میانی این اجازه را می دهند که بسته هایTCP را نگهداری کنند و بسته های داده را از نودی به سمت نود دیگر بفرستند.

DTC : بسته های داده TCP را cache کرده و بلافاصله بسمت جلو هدایت می کند. در حالیکه

TSS: بسته cache شده را تا زمانیکه از دریافت موفقیت آمیز بسته قبلی توسط نود بعدی مطلع نشود , آنرا بسمت جلو هدایت نمی کند.

هر دو دیدگاه بطور آشکار تعداد بسته های TCP و Ack را کاهش می دهند.و کارایی آنها از نقطه نظر نرخ خطا تا حدی با یکدیگر اختلاف دارد و این دو مکانیزم نیازهای متفاوتی را به لحاظ مصرف حافظه بافر و تنظیمات TCP دارا می باشند.

شبکه های حسی بیسیم از تعداد زیادی حسگر های حساس به امواج رادیویی تشکیل شده اند که بطور خودکار تشکیل یک شبکه را می دهند که داده ها در بین حسگرها منتقل می شوند. قطعات استفاده شده با مشکل محدودیت منابع از جمله انرزی, قدرت پردازشگر , حافظه و پهنای باند ارتباطی روبرو می باشند.

برنامه های زیادی در شبکه حسگرهای بیسیم نیازمند ارتباطات خارجی به منظور کنترل و مانیتور کردن موجودیتهایی هستند که از داده های موجود درشبکه استفاده می کنند و با حسگرها در ارتباط هستند. اجرای TCP/IP در شبکه های حسی ارتباط مستقیم شبکه حسی را با شبکه مبتنی بر IP به راحتی و بدون نیاز به واسطه فراهم می سازد.

از آنجاییکه هر حسگر قادر به برقراری ارتباط با استفاده از TCP/IP می باشد, مسیریابی داده ها از شبکه های حسی با استفاده از تکنولوزی مبتنی بر IP مثل GPRS ممکن است.

انتقال داده ها در شبکه حسی مبتنی بر IP با استفاده از دو پروتکل مهم امکانپذیر می شود: پروتکل UDP که best effort است و پروتکل TCP که جریان مطمئن بایتهای داده را حمایت می کند. UDP برای داده ها و اطلاعات دیگری که در جریان انتقال, خیلی اهمیت ندارند استفاده می شود . اما در مورد داده های مهم از پروتکل TCP استفاده می شود. به عنوان مثال پیکره بندی و مانیتورینگ حسگرها و ... مخصوصا بارگذاری داده ها در نودهای مخصوصی مثل نود ریشه در یک ساختار کلاستری در یک ناحیه مخصوص.

نکته مهم در استفاده از TCP مشکل اساسی آن به لحاظ کارایی در یک شبکه بیسیم است . یک مساله این است که TCP که برای شبکه های با سیم و تعداد خطای پایین طراحی شده , از اتلاف بسته ها بعنوان نشانه ای از ازدحام در شبکه و در نتیجه کاهش نرخ ارسال استفاده می کند. که این امر منجر به کاهش بازدهی می شود. که مشکل بزرگی در شبکه AD-HOC می باشد.

برای شبکه های حسی مشکل اصلی, ناکارآمدی انرزی در استفاده از TCPاست . این ناکارآمدی انرزی بعلت انتقال انتها به انتهای بسته های داده در TCP است, که در مورد بسته های تلف شده نیاز به انتقال مجدد از حسگر فرستنده است. در شبکه با چندین نود (پرش) ارسال مجدد بسته ها , باید از کلیه نودهای میانی بین فرستنده و گیرنده بسمت جلو انجام شود که این امر منجر به اتلاف انرزی زیادی در هر پرش می شود. در نتیجه از آنجا که در هر پرش , نرخ اتلاف بسته ها در بازه ۵٪ تا ۱۰٪ و در مواردی بیشتر نیز می

باشد ارسال انتها به انتهای بسته های تلف شده روش مناسبی برای انتقال قابل اطمینان در شبکه های حسی نمی باشد. در این جا دو تکنیک DTC و TSS شرح داده می شوند. در هر دو مکانیزم، نودهای میانی بسته های TCP را cache کرده و در هنگام اتلاف بسته ها بصورت محلی ارسال مجدد توسط این نودها صورت می گیرد. و پیاده سازی این دو تکنیک نیاز به هیچگونه تغییری در پیاده سازی TCP ندارد.

در مکانیزم TSS ,نودی که بسته ای را cache کرده بسته را بسمت جلو هدایت نمی کند تا زمانیکه از دریافت تمامی بسته های TCP با شماره ترتیبی کمتر توسط نود بعدی خود آگاه شود. این امر منجر به وجود back pressure می شود که تعداد بسته های ارسالی را کاهش می دهد.

DTC بلافاصله همه بسته های TCP را بسمت جلو هدایت می کند , حتی آنهایی که در نودهای میانی cache شده اند.

بعلاوه TSS از مکانیزم بهبود تصدیق (Acknowledge) TCP استفاده می کند.

نتایج بدست آمده نشان می دهند که هر دو مکانیزم TSS و DTC بطور آشکاری کارایی TCP را در قالب تعداد کل سگمنتهای منتقل شده و تعداد انتقالهای مجدد انتها به انتها افزایش می دهند. با وجود مکانیزم Ack تهاجمی در TSS تعداد کل Ack بیشتر از DTC می باشد. بعبارت دیگر , بدلیل وجود مکانیزم back pressure تعداد سگمنتهای منتقل شده در TSS کمتر می باشد.

برای نرخ پایین اتلاف بسته ها, DTC و TSS تعداد بسته های منتقل شده مشابهی دارند. در حالیکه DTC نسبت به TSS بسته های بیشتری را در شرایط نرخ اتلاف بالا ، بازیابی می کند.

۲-۲)شبکه های حسی مبتنی بر IP:

علاوه بر موضوع کارایی انرزی TCP ,چندین مسئله وجود دارد که باید قبل از آنکه TCP/IP بصورت کارآمد در شبکه های حسی بیسیم مورد استفاده قرار گیرد, بررسی و رفع گردد. که در اینجا به بررسی این موارد می پردازیم.

۱-۲-۲)محدودیت گره ها: به منظور انعطاف پذیری در شبکه های حسی، هر نود حسگر بطور نمایان به لحاظ حافظه و قدرت پردازش محدود می باشد. در نتیجه چنین تصور می شود که پشته TCP/IP برای چنین سیستم کوچکی بسیار سنگین می باشد

۲-۲-۲)آدرس دهی مرکزی: آدرسهای IP در شبکه های IP قدیمی بر مبنای توپولوژی شبکه نسبت داده می شد. یک آدرس IP منحصر بفرد بطور دستی یا از طریق مکانیزمهای اتومات از قبیل DHCP اختصاص داده می شد. این مکانیزمهای آدرس دهی در مورد شبکه های حسی وسیع کاربرد ندارند. به جای آن در شبکه های حسی مبتنی بر

IP از نگاشت آدرس IP سه بعدی استفاده می شود. که از مکان گـره هـای حسـی بـرای ساختن آدرس IP استفاده می کند.

۳-۲-۲)مسیر یابی متمرکز:در شبکه های IP اصلی ، بسته ها در طول شبکه جریان دارند. مسیر هر بسته به آدرسهای IP و توپولوژی شبکه بستگی دارد. برای شـبکه هـای حسـی بیسیم، مکانیزم مسیر یابی متمرکز داده اغلب ترجیح داده میشود. برای پیاده سازی مسیر یابی متمرکز داده ها در شبکه های حسی مبتنی بر IP ، از برنامـه هـای کـاربردی (کـه شبکه را تحت پوشش قرار می دهند) استفاده می شود.

۴-۲-۲)سر بار هدر:در مقایسه با پروتکلهای خاص شبکه های حسـی، پروتکلهـا در شـبکه های TCP/IP سر بار هدر زیادی دارند.

خصوصیات مشترک ذاتی در شبکه های حسی نیاز به ساختار هدر کارآمدی دارد.

۳-۲)Distributed TCP Caching:

ایده اصلی مکانیزم DTC، اجتناب از انتقال مجدد انتها به انتهاست، بدین ترتیـب کـه در هنگام وقوع اتلاف بسته ها ،بسته هـای TCP در داخـل شـبکه cache شـده و بصـورت محلی ارسال مجدد آنها انجام می شود. حالـت ایـده ال آن اسـت کـه هـر گرهـی، تمـام

سگمنتها را cacheکرده و دقیقا عملیات ارسال مجـدد بسـته هـا از آخـرین گرهـی کـه سگمنت را قبل از اتلاف آن ارسال کرده، صورت پذیرد.

هر چند که، بدلیل وجود محدودیت در منابع حسگرها، فرض بر این است کـه هـر گرهـی تنها می تواند یک سگمنت را نگهداری نماید و از سـگمنت cache شـده نگهـداری ویـژه داشته باشد ، با این فرض که ممکن است توسط گره بعدی دریافت نشود. DTC فقط در گره های حسگرهای میانی پیاده سازی می شود و نیازی به انجام تغییرات در حسـگرهای انتهایی نیست.

یک حسگر بعنوان گیرنده ممکن است از جنبه های TCP بصورت زیر استفاده نماید:

گیرنده کوچکترین حداکثر سایز سگمنت را به منظور اجتناب از زیـاد پـر کـردن ظرفیـت گره های حسگر اعلان می کند . بعلاوه، گیرنده یک سایز پنجـره کوچـک را کـه نشـان از تعداد سگمنتهای در جریان دارد را اعلان می کند.

۱-۳-۲)مکانیزمهای پروتکل:

بدلیل محدودیت در حسگرها، روش مناسب بـرای نودهـا در انتخـاب سـگمنتی کـه بایـد cache شود در کارایی DTC بسیار تاثیر گذار است. حالت مطلـوب بـرای انتخـاب ایـن است که تمامی سگمنتها که در حال جریان در شبکه هستند توسط نود cache شـوند و از آنجاییکه احتمال دارد سگمنتهای cache شده در مسیر به سمت مقصد دچـار مشـکل شوند از آنها نگهداری شود. بـرای رسـیدن بـه ایـن هـدف، نودهـا، سـگمنتهای TCP بـا

بالاترین شماره را قطعا cache می کنند. این امر، cache شدن سگمنهای قدیمی را در شبکه تضمین می کند.

به منظور شناسایی اتلاف بسته در نود (پرش) بعدی از Ack تولید شده از فیدبک لایه پیوند داده ها استفاده می شود.

این روش همچنین با مکانیزم استراق سمع نیز عمل می نماید.زمانیکه یک نود متوجه ارسال یک بسته توسط نود بعدی خود می شود، یک سگمنت TCP که به سمت جلو هدایت شده ولی در ازای آن هیچ Ack از لایه پیوند برگشت نخورده ممکن است در انتقال از بین رفته باشد . بنا براین، این سگمنت که cache شده بود قفل می شود تا بوسیله سگمنت دیگری با شماره بزرگتر بازنویسی نشود. یک سگمنت قفل شده تنها زمانی از cache خارج می شود که Ack مبنی بر دریافت ان برسد یا زمانیکه Time Out اتفاق بیفتد.

۲-۳-۲)شناسایی اتلاف بسته ها و ارسال مجدد بصورت محلی:

به منظور اجتناب از ارسال مجدد انتها به انتها، DTC نیاز دارد که نسبت به TCP بطور سریعتر به اتلاف بسته ها پاسخ دهد.

DTC بطور عمده از طریق وقوع Time Out متوجه اتلاف بسته ها می شود. با این فرض که مسیرها هم اندازه و متناسب باشند ، نودها می توانند تاخیر بین نود و نقطه انتهایی را تخمین بزنند. هر نودی میزان زمان یک رفت و برگشت به سمت گیرنده (rtt) را اندازه گیری می نماید و مقدار Time Out را ۱,۵*rtt تنظیم می نماید.

این مقدار تضمین می کند که مقدار Time Out کوچکتر برای نودهای نزدیک به مقصد و برای نودهای نزدیک به مبدا بزرگتر است. از آنجا که مقدار rtt واقعی اغلب کوچکتر از میزان تخمین زده شده توسط نقاط انتهایی TCP است و نودهای میانی قادر به ارسال مجدد بسته ها زودتر از نقاط انتهایی TCP می باشند. نودهای DTC در هنگام ارسال مجدد محلی سگمنت cache شده (قفل شده) یک تایمر را فعال می نمایند و شبیه سازی نشان داده است که مکانیم استاندارد شده دیگری به منظور شناسایی اتلاف بسته ها، Ack تکراری تولید کرده و به لحاظ کارایی DTC مفید نمی باشد.

۲-۳-۳)Selective Acknowledgement:

DTC از مفهوم TCP SACK به منظور شناسایی اتلاف بسته ها و همچنین مکانیزم سیگنال دهی بین نودها استفاده می کند. به منظور آگاه ساختن سایر نودها از سگمنتهای قفل شده در cache از روشهای زیر استفاده میکند. با رسیدن یک تصدیق با شماره کوچکتر از شماره ترتیبی سگمنت cache شده یک نود عملیات زیر را انجام میدهد:

اگر شماره ترتیبی سگمنت cache شده در یک نود، در بلاکSACK نباشد، نود سگمنت cache شده را مجددا ارسال می کند. قبل از انتقال Ack به سمت فرستنده، نود شماره ترتیبی سگمنت cache شده را به بلاک SACK اضافه میکند و اگر سگمنتهای cache شده تمام فضای خالی را اشغال کنند (با cache کردن تمام شماره سگمنتها تا بزرگترین شماره در بلاکSACK کهAck داده شده)، نود می تواند

تصدیق را از بین ببرد. ذکر این نکته لازم است که نود نباید یک تصدیق جدید برای تصدیق دادن تمام سگمنتها در بلاکهای SACK تولید کند، از آنجا که گیرنده اجازه صرفنظر کردن از سگمنت SACK شده قبلی را دارد.

یک نود می تواند حافظه خود را خالی کند اگر شماره ترتیبی سگمنتهایcache شده در بلاکSACK باشد و این امر بدین مفهوم است که یا گیرنده سگمنتهای مورد نظر را دریافت کرده و یا سگمنتها بوسیله یک نود نزدیکتر به گیرنده cache شده و قفل شده اند.

یادآوری می شود حتی اگر فرستنده از مکانیزم SACK حمایت نکند ، مکان اصلی یا اولین نود در شبکه حسی ممکن است حالتهای SACK را اضافه یا کم کند تا سیگنال دهیSACK برایDTC فعال شود.

۴-۳-۲)تولید مجدد تصدیق بصورت محلی:

در حالیکهDTC بسته هایTCP را cache می کند تصدیق بسته ها راcache یا ارسال مجدد نمی کند. در واقع DTC از مکانیزم تولید مجدد محلی تصدیق استفاده می کند. زمانیکه یک نود میانی یک بسته دادهTCP را می بیند که برای آن قبلا تصدیق دریافت و هدایت شده است، نود فرض را بر این می گذارد که تصدیق بسته گم شده است. بنابراین، سگمنت داده را به جلو ارسال نمی کند بلکه در عوض بصورت محلی

تصدیق آنرا تولید می کند. که این مسئله می تواند بدون هیچگونه عملیات بافرینگ یا cache کردن تصدیق اولیه صورت پذیرد.

۴-۲) TCP Support for Sensor Nodes :

TSS(TCP Support for Sensor Nodes) لایه ای بین TCP و لایه مسیر یابی است که که در پشته پروتکل مسیریابی در نودهای حسگر پیاده سازی میشود و هدف آن انجام عملیات با مصرف بهینه انرزی در نودهای حسگر می باشد.

TSS باید در نودهای حسگر از فرستنده تا گیرنده و نودهای میانی که بستر عبور سگمنتها و تصدیق در یک ارتباط TCP می باشند, پیاده سازی شود.

همانند TSS, DTC نیاز به ذخیره اطلاعات وضعیت برای هر ارتباط TCP را دارد.

اطلاعات وضعیت کمتر از ۲۰ عدد هستند و بطور عمده شامل شماره تائید و شماره ترتیبی و rtt تخمینی می باشند.

TSS به منظور کاهش تعداد انتقالات از مفاهیم و مکانیزمهای زیر استفاده می کند:

- بسته هایی که ممکن است توسط نود بعدی دریافت نشوند را cache می کند.

این بسته ها براساس مکانیزم استراق سمع یا تصدیق TCP شناخته می شوند.

- ارسال مجدد سگمنتهای TCP بصورت محلی براساس تخمین rtt.
- تولید مجدد TCP Ack و بازیابی براساس تخمین انتشار تاخیر و استراق سمع.

- استفاده از مکانیزم Backpressure به منظور اجتناب از هدایت بسته ای به سمت نود بعدی در حالیکه بسته های قبلی توسط آن نود دریافت نشده باشد.

این مکانیزم بطور کلی براساس سگمنتها ی TCP و تصدیق TCPعمل می نماید. در واقع سگمنتهای TCP و تصدیق TCPتنها بسته هایی مورد نیاز می باشند.این دیدگاه علاوه بر اینکه تعداد انتقالات را کاهش میدهد می تواند بر روی لایه MAC در هر نوع شبکه حسی مورد استفاده قرار گیرد.

TSS همچنین تضمین می کند که بسته ها بترتیب به مقصد می رسند. که این امر از بافرینگ مجدد بدلیل عدم رعایت ترتیب و تائیدیه انتخابی در TCP جلوگیری می کند.

۱-۴-۲)مکانیزمهای پروتکل:

مشابه TSS، DTC به نودهای میانی اجازه cache کردن سگمنتهای TCP را میدهد. هر چند که، برخلاف DTC تصمیم گیری به منظور cache کردن سگمنتهای کاملا منطقی است. یک نود همیشه سگمنتی را cache می کند که داده های آن هنوز Ack دریافت نکرده اند یا بوسیله نود بعدی هدایت نشده اند. یک بسته در cache می ماند تا زمانیکه این اطمینان حاصل شود که سگمنت توسط نود بعدی (به سمت مقصد) دریافت شده است. یک نود زمانی از دریافت سگمنت توسط نود بعدی آگاه می شود که متوجه هدایت سگمنت توسط نود بعدی شود یا زمانیکه Ackبسته از سمت گیرنده به فرستنده ارسال شود. در اینحالت فرض می شود که به ارسال بسته ها توسط نودهای همسایه

گوش می دهند تا بتوانند تشخیص دهند که آیا نود همسایه سگمنت را بـه جلـو هـدایت نموده است یا نه. و زمانیکه نود از دریافت سگمنت توسط نود بعدی آگاه شد این سگمنت از cache خارج می شود. مشابهDTC درcache تنها یک بسته نگهداری می شود. هر چند که، مکانیزم منتخب برای TSS نیاز به بافر دیگری به منظـور ذخیـره موقـت بسـته هایی که منتظر ارسال توسط نود بعدی می باشند. برای بسته ها دارد.

۲-۴-۲)انتقال مجدد سگمنتها یTCP بصورت محلی:

تمام نودهای میانی قادر به ارسال مجدد بسته ها بصورت محلی هستند. که این امر زمـانی صورت می گیرد که نودها فرض کنند که سگمنت cache شده توسط نود بعدی دریافـت نشده است. ارسال مجدد بطور عمده با وقوع Time Out شروع می شود. کـه در نتیجـه نیاز به تنظیم دقیق میزان Time Out می باشـد. همـانطور کـه میـزان Time Out در ارسال مجدد ۱,۵*rtt تنظیم می شود. شبیه سازی نشان مـی دهـد کـه تنظـیم Time Outدر ارسال مجدد به میزان ۲* rtt در مواردی بد عمل می نماید . اما تنظـیم Time Out در ارسال مجدد به اندازه ۱,۵*rtt اجازه عکس العمل سریعتر نسبت به اتلاف بسـته ها را و جبران اتلاف چندین بسته را قبل از اینکه Time Out انتها به انتها فعال شـود را می دهد.

میزان Time Out که از مقصد به سمت مبدا در حال افـزایش اسـت همچنـین در خـور نیاز مکانیزم bachpressure می باشد که در ادامه توضیح داده می شود.

یک مسئله مورد بحث این است که اجبار نودهای حسگر بـه اسـتراق سـمع بسـته هـا بـه لحاظ مصرف انرژی کارامد نیست. به عبارت دیگر، نود جدید فقط باید به ارسـالات دیگـر نودها توجه کند برای مدت زمان کوتاهی.

به طور نمونه، یک بسته توسط نود بعدی بلافاصله به جلو هـدایت خواهـد شـد و فقـط در حالت اتلاف بسته یک نود باید برای تمامی ارسالات مجدد مـدت Time Out را اسـتراق سمع کند.

مسئله دیگر می تواند این باشد که نود بعدی با قدرت کمتری عملیات انتقال را انجام مـی دهد که در نتیجه نود قبلی نمی تواند در مورد بسته ارسالی استراق سمع کنـد. هـر چنـد که، ممکن است تطبیق قدرت ارسال هم به سمت نود بعدی و هم نود قبلی و انتقال یـک سگمنتTCP نسبت به ارسال دو سگمنت TCP به سمت نود بعدی و ارسال Ackبـه نود قبلی با میزان انرژی تطبیق شده ، کارآمدتر باشد.

علاوه بر این، ممکن است این اتفاق بیفتد که Time Out ارسال مجدد یک نود به پایـان برسد و نود سگمنت را مجددا ارسال کند اگر چه که یـک بسـته(سـگمنت) توسـط نـود بعدی دریافت شده و به سمت جلو هدایت شده است. این امر ممکن است اتفـاق بیفتـد ، اگر نود قبلی یک هدر بسته با خطا دریافت کند و Ack را رها کند. پس سـگمنت TCP که قبلا بدرستی ارسال شده نباید مجددا ارسـال شـود . ارسـال بایـد توسـط یـک فیلتـر جلوگیری شود که فیلترها تمامی سگمنتهایی را که قبلا به جلو ارسال شـده انـد را عبـور می دهند. البته، یک ارسال مجدد انتها به انتها نباید فیلتر شود، زیرا انتقال مجدد انتها به

انتها ممکن است بدلیل بازیابی مسئله مهم مثل نقص مسیر باشد کـه نـود بعـدی ممکـن است ازآن اگاه نباشد. سگمنتهای TCP که مجددا ارسال شده اند بوسـیله آدرس مبـدا و فیلد شناسایی IP قابل تشخیص هستند.

۳-۴-۲)تولیـد مجـدد و بازیـابی تصـدیق (TCP Acnowledge) : تصـدیق TCP در TSS بسیار مهم است. از آنجا کـه چنـدین مکـانیزم از جملـه تخمـین rtt، و caching ارسال مجدد به آن وابسته می شود. تحقیقات نشان می دهد که اتـلاف تصـدیقها (Ack) ممکن است تاثیر بسیاری بر روی تعداد سگمنتهای انتقالی TCP داشـته باشـد. TSS از دو مکانیزم برای ارسال مجدد تصدیق TCP استفاده می کنـد کـه ایـن مکانیزمهـا بطـور آشکار باعث کاهش تعداد سگمنتهای ارسالی می شوند:

مکانیزم تولید مجدد محلی همانگونـه کـه در DTC انجـام مـی شـود و مکـانیزم بازیـابی تصدیق به صورت تهاجمی، که در صورتیکه ارسال آنها توسـط نـود بعـدی تشـخیص داده شود، دوباره تصدیق را بازیابی می کند.

از آنجا که تصدیق TCP باید معمولا بدون تاخیر فاحش به سمت مبدا ارسال شـود ، هـر نود زمان بین ارسال تصدیقTCP و ارسال تصـدیق از نـود بعـد ی بـه سـمت فرسـتنده سگمنت را اندازه گیری می نماید. مشابه تخمین rtt ، از رویه میانگین گیری استفاده مـی نماییم و Time Out تصدیقTCP را به میزان دو برابر میانگین تنظیم می نماییم. بعـد از سرآمدن Time Out ، تصدیقTCP باز یابی می شود. با استفاده از بـالاترین شـماره

تصدیق. دوباره این امر نیاز به cache کردن تصدیق ندارد اما تصدیق ها می توانند با استفاده از اطلاعات وضعیت بازیابی شوند.

۴-۴-۲)مکانیزم Back pressure: اگر نود بعدی (جانشین یک نود) تمامی بسته های دریافتی را ارسال نکند باعث وجود مشکل در شبکه می شود. برای مثال ، در شبکه ممکن است ازدحام رخ دهد یا هدایت بسته ها پیشرفتی نداشته باشد. زیرا که سگمنت TCP با بیت خطا در ابتدا نیاز به بازیابی دارد. اگر یک نود به ارسال بسته ها به این طریق ادامه دهد، احتمال ارسال غیر لازم بسته ها بالا می رود. در شرایط ازدحام ، یک سگمنت ارسالی به راحتی ممکن است گم شود. بطور مشابه، اتلاف بسته ها که منجر به بیت خطا می شود صادق است.

زیرا در چنین شرایطی تمام حافظه های cache در نودها پر هستند و انتقال یک بسته جدید حفاظت نمی شود. به این دلیل، درTSS یک نود هرگونه ارسالی را متوقف می سازد تا زمانیکه تمامی بسته های قبلی دریافت شده باشند و توسط نود بعدی به جلو ارسال شده باشد. اگر ارسال بسته ها به دلایلی متوقف شد، بر روی دیگر نودها در زنجیره قبل از نود متوقف شده تاثیر می گذارد. این بدان مفهوم است که تمامی نودهای دیگر همچنین ارسال خود را متوقف می کنند تا زمانیکه پیشروی در نود بعدی مربوطه اتفاق بیفتد.

در موارد اتلاف بسته (بدلیل وجود ازدحام یا خطاهای بیتی) اتلاف بسته باید توسط نـودی که بسته را آخرین بار فرستاده بازیابی شود. در این مـوارد، بایـد از شـروع ارسـال مجـدد توسط نودهای قبل از نودی که باید عملیات بازیابی را انجام دهد، جلوگیری نمـود، یعنـی نودهای نزدیکتر به فرستنده.

بنابراین باید میزان Time Out ارسال مجدد را برای نودهای نزدیک به فرستنده افـزایش داد. برای این منظور، مکانیزم این اطمینان را می دهد Time Out که ارسـال مجـدد در طول نودها از سمت گیرنده به سمت فرستنده افزایش می یابد همانگونه که در بخـش ۲-۴-۲ شرح داده شد.

منابع:

[1] Jayakrishnan Mundarath, Narasimhan Venkataramaiah, Ronald Huang , TCP Variants Simulation –based Analysis Multiple Implementations and Features Comparison Report.

[2] RFC 2581 TCP Congestion Control

[3] Juniper NETWORK ,Inc. Supporting Differentiated Service Classes: TCP Congestion Control Mechanisms.

[4] Lawrence S. Brakmo, Sttudent Member , IEEE , and Larry L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet.

[5] U .Hengartner , J.Bolliger and Th. Gross. TCP Vegas Revisited.

[6] Jacobson , V." Congestion Avoidance and Control ", Computer Communication Review vol 18 , no.4 ,pp314-329,August1988

[7] Thiemo Voigt, Adam Dunkels. Energy-Efficient TCP Operation in Wireless Sensor Networks.

[8] Torsten Braun1, Thiemo Voigt2, and Adam Dunkels. TCP Support for Sensor Networks

[9] Adam Dunkels, Juan Alonso, Thiemo Voigt .
Making TCP/IP Viable for Wireless Sensor Networks

[10] Adam Dunkels, Juan Alonso, Thiemo Voigt . Distributed TCP Caching for Wireless Sensor Networks